ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προσομοιωτές χρονοπρογραμματισμού πραγματικού χρόνου Λειτουργικά συστήματα πραγματικού χρόνου

Ευάγγελος Π. Παπαδόπουλος

Επιβλέποντες : **Ευστάθιος Χατζηευθυμιάδης,** Επίκουρος Καθηγητής
Λάζαρος Μεράκος, Καθηγητής

ΑΘΗΝΑ

ΔΕΚΕΜΒΡΙΟΣ 2011

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προσομοιωτές χρονοπρογραμματισμού πραγματικού χρόνου
Λειτουργικά συστήματα πραγματικού χρόνου

Ευάγγελος Π. Παπαδόπουλος

Α.Μ.: 1024

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ευστάθιος Χατζηευθυμιάδης,** Επίκουρος Καθηγητής
Λάζαρος Μεράκος, Καθηγητής

ΔΕΚΕΜΒΡΙΟΣ 2011

ΠΕΡΙΛΗΨΗ

Στην εργασία αυτή αφού γίνει μία σύντομη περιγραφή των συστημάτων πραγματικού χρόνου μελετώνται οι αλγόριθμοι χρονοπρογραμματισμού για συστήματα πραγματικού χρόνου. Γίνεται η περιγραφή προσομοιωτών χρονοπρογραμματισμού που έχουν στόχο την έρευνα και την εκπαίδευση και παρέχονται δωρεάν. Από αυτούς περιγράφετε εκτενώς η χρήση του προσομοιωτή Cheddar. Τέλος περιγράφονται τα λειτουργικά συστήματα πραγματικού χρόνου και γίνεται αναλυτική περιγραφή των προτύπων και κάποιων υλοποιήσεων τόσο από το χώρο του ελεύθερου λογισμικού όσο και από τον χώρο του εμπορικού λογισμικού.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Λειτουργικά συστήματα.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Cheddar, Real time operating system, RTOS, Real-Time Scheduling, Scheduling Simulation, προσομοιωτές χρονοπρογραμματισμού, χρονοπρογραμματισμός πραγματικού χρόνου, λειτουργικά συστήματα πραγματικού χρόνου.

ABSTRACT

In this paper after a brief description of the real-time systems we study scheduling algorithms for real-time systems. We study a number of scheduling simulators aimed at research and education area and are free of any charge. We study in depth the Cheddar simulator. Finally we describe the real time operating systems and give detailed description of standards and some implementations from the open software community and from the commercial software.

SUBJECT AREA: Operating Systems.

KEYWORDS: Cheddar, Real time operating system, RTOS, Real-Time Scheduling, Scheduling Simulation.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή εκπονήθηκε στο πλαίσιο του προγράμματος μεταπτυχιακών σπουδών του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Θα ήθελα να ευχαριστήσω όσους συνέβαλαν στην εκπαίδευση μου και ιδιαίτερα τον κ. Ευστάθιο Χατζηευθυμιάδη για την συνεργασία μας.

1. ΕΙΣΑΓΩΓΗ

Από την δεκαετία του 1950 που εμφανίστηκαν για πρώτη φορά, τα λειτουργικά συστήματα συνεχώς εξελίσσονται. Ακολουθούν την διαρκή και έντονη εξέλιξη της αρχιτεκτονικής και του υλικού των υπολογιστών, σύμφωνα με τον νόμο του Moore η ισχύς του υλικού διπλασιάζεται κάθε 18 μήνες, γιατί παρά τις αφαιρέσεις , το λειτουργικό σύστημα είναι στενά συνδεδεμένο με την αρχιτεκτονική του υποκείμενου υλικού.

1.1 Στόχοι της εργασίας.

Μία, από τις πολλές κατηγορίες λειτουργικών συστημάτων, είναι τα λειτουργικά συστήματα πραγματικού χρόνου που υποστηρίζουν συστήματα πραγματικού χρόνου. Σήμερα, τα συστήματα πραγματικού χρόνου διαδραματίζουν κρίσιμο ρόλο στην κοινωνία μας γιατί βρίσκουν πεδίο εφαρμογής σε πολλού τομείς, το πεδίο των εφαρμογών τους δε συνεχώς διευρύνεται.

Οι στόχοι αυτής της εργασίας είναι:

1. Να γίνει περιγραφή των συστημάτων πραγματικού χρόνου.
2. Να παρουσιαστούν εργαλεία προσομοίωσης χρονοπρογραμματισμού πραγματικού χρόνου που μπορούν να χρησιμοποιηθούν σαν εκπαιδευτικά εργαλεία ώστε οι φοιτητές να έχουν μια ευρύτερη και βαθύτερη κατανόηση των αποτελεσμάτων της θεωρίας του χρονοπρογραμματισμού πραγματικού χρόνου.
3. Να γίνει περιγραφή των προτύπων για λειτουργικά συστήματα πραγματικού χρόνου και εκτενής παρουσίαση υλοποιήσεων λειτουργικών συστημάτων πραγματικού χρόνου.
4.

1.2 Δομή της εργασίας.

Κεφάλαιο 1. Εισαγωγή.
Σε αυτό το κεφάλαιο περιγράφουμε τους στόχους και τη δομή της εργασίας.

Κεφάλαιο 2. Συστήματα πραγματικού χρόνου.
Σε αυτό το κεφάλαιο περιγράφουμε τις βασικές έννοιες και τα χαρακτηριστικά των συστημάτων πραγματικού χρόνου και στη συνέχεια τις βασικές έννοιες και τα χαρακτηριστικά των λειτουργικών συστημάτων πραγματικού χρόνου. Στην πρώτη παράγραφο περιγράφουμε τις βασικές έννοιες των συστημάτων πραγματικού χρόνο. Στη δεύτερη παράγραφο κάνουμε μία σύντομη καταγραφή των σημαντικών όρων των λειτουργικών συστημάτων με έμφαση στα πραγματικού χρόνου. Στην τρίτη παράγραφο αναφέρουμε τα χαρακτηριστικά ενός λειτουργικού συστήματος πραγματικού χρόνου και τέλος στην τέταρτη παράγραφο περιγράφουμε πως μπορεί κανείς να επιλέξει ένα RTOS.

Κεφάλαιο 3. Χρονοπρογραμματισμός πραγματικού χρόνου.
Σε αυτό το κεφάλαιο περιγράφουμε τις βασικές έννοιες του χρονοπρογραμματισμού πραγματικού χρόνου, τους αλγορίθμους χρονοπρογραμματισμού περιοδικών έργων και θα παρουσιάσουμε προσομοιωτές χρονοπρογραμματισμού πραγματικού χρόνου. Η οργάνωση αυτού του κεφαλαίου είναι η ακόλουθη: Στην πρώτη παράγραφο αναφέρουμε τις βασικές έννοιες και τις παραμέτρους του χρονοπρογραμματισμού πραγματικού χρόνου. Στη δεύτερη παράγραφο περιγράφουμε τέσσερεις βασικούς αλγορίθμους. Στην τρίτη παρουσιάζουμε προσομοιωτές χρονοπρογραμματισμού πραγματικού χρόνου και

τέλος στην τέταρτη παράγραφο δίνουμε παραδείγματα από τη χρήση του προσομοιωτή Cheddar.

Κεφάλαιο 4. Λειτουργικά συστήματα πραγματικού χρόνου.

Σε αυτό το κεφάλαιο θα περιγράψουμε τα χαρακτηριστικά των κυριότερων λειτουργικών συστημάτων πραγματικού χρόνου. Στην πρώτη παράγραφο έχουμε συμπεριλάβει τα πρότυπα POSIX, OSE και μTRON. Στη δεύτερη παράγραφο το εμπορικό λογισμικό OSE, OSEK/VDX και QNX. Στην τρίτη παράγραφο το λειτουργικό σύστημα eCOS από το χώρο του ανοικτού λογισμικού και πως το Linux τροποποιήθηκε για να υποστηρίζει εφαρμογές πραγματικού χρόνου.

Η ιστοθέση των εταιρειών που αναπτύσσει και υποστηρίζει το κάθε προϊόν και ήταν η κύρια πηγή πληροφοριών αναγράφεται σε παρένθεση δίπλα στο RTOS που περιγράφουμε. Στο παράρτημα Ι υπάρχει λεπτομερής πίνακας με τις διευθύνσεις εταιρειών και τα προϊόντα τους.

Κεφάλαιο 5. Συμπεράσματα..

Σε αυτό το κεφάλαιο βρίσκονται τα συμπεράσματα της εργασίας.

Στο συνοδευτικό CD βρίσκετε η εργασία σε ηλεκτρονική μορφή doc και pdf.

2. Συστήματα πραγματικού χρόνου.

Σε αυτό το κεφάλαιο θα περιγράψουμε τις βασικές έννοιες και τα χαρακτηριστικά των συστημάτων πραγματικού χρόνου και στη συνέχεια τις βασικές έννοιες και τα χαρακτηριστικά των λειτουργικών συστημάτων πραγματικού χρόνου. Στην πρώτη παράγραφο περιγράφουμε τις βασικές έννοιες των συστημάτων πραγματικού χρόνο. Στη δεύτερη παράγραφο κάνουμε μία σύντομη καταγραφή των σημαντικών όρων των λειτουργικών συστημάτων με έμφαση στα πραγματικού χρόνου. Στην τρίτη παράγραφο αναφέρουμε τα χαρακτηριστικά ενός λειτουργικού συστήματος πραγματικού χρόνου και τέλος στην τέταρτη παράγραφο περιγράφουμε πως μπορεί κανείς να επιλέξει ένα RTOS.

2.1 Εισαγωγή.

Συστήματα πραγματικού χρόνου (Real Time Systems) είναι υπολογιστικά συστήματα που πρέπει να αντιδράσουν μέσα σε συγκεκριμένες χρονικές προθεσμίες όταν δεχτούν κάποιο σήμα εισόδου από το περιβάλλον τους. Κατά συνέπεια, η σωστή συμπεριφορά τους δεν εξαρτάται μόνο από την αξία των αποτελεσμάτων των υπολογισμών, αλλά και από το χρόνο κατά τον οποίο αυτά παράγονται [1]. Ένα αποτέλεσμα που υπολογίζεται πολύ αργά μπορεί να είναι άχρηστο ή ακόμη και επικίνδυνο. Σήμερα τα συστήματα πραγματικού χρόνου διαδραματίζουν κρίσιμο ρόλο στην κοινωνία μας γιατί βρίσκουν πεδίο εφαρμογής σε πολλού τομείς όπως: στον έλεγχο της παραγωγικής διαδικασίας πυρηνικής ενέργειας, στον έλεγχο της παραγωγικής διαδικασίας σε εργοστάσια χημικών προϊόντων, στον έλεγχο πολύπλοκων διαδικασιών παραγωγής, στα συστήματα σιδηροδρομικών μεταφορών, εφαρμογές στα αυτοκίνητα, στα συστήματα ελέγχου πτήσης αεροσκαφών, στην συλλογή περιβαλλοντικών δεδομένων και στην παρακολούθησή τους, στα τηλεπικοινωνιακά συστήματα, στα ιατρικά συστήματα, στους βιομηχανικοί αυτοματισμοί, στη ρομποτική, στα στρατιωτικά συστήματα, στις διαστημικές αποστολές, στις καταναλωτικές ηλεκτρονικές συσκευές, στα συστήματα πολυμέσων, στα έξυπνα παιχνίδια, στην εικονική πραγματικότητα.

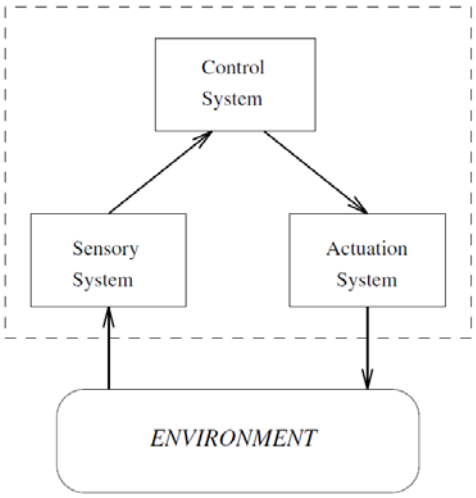
Σε πολλές περιπτώσεις, το σύστημα που εκτελεί τους υπολογισμούς πραγματικού χρόνου είναι ενσωματωμένο (embedded) στο σύστημα που πρέπει να ελέγχει. Τα ενσωματωμένα συστήματα εκτείνονται από τις μικρές φορητές συσκευές, όπως τα κινητά τηλέφωνα, οι φωτογραφικές μηχανές μέχρι μεγάλα συστήματα όπως, τα βιομηχανικά ρομπότ, τα αυτοκίνητα και τα αεροπλάνα.

Πολλοί πιστεύουν ότι γράφοντας τον περισσότερο κώδικα μιας εφαρμογής σε γλώσσα μηχανής επιτυγχάνουν τους στόχους τους. Αυτή η προσέγγιση είναι λανθασμένη γιατί έχει τα μειονεκτήματα του επίπονου προγραμματισμού, του δυσνόητου κώδικα, το λογισμικό είναι δύσκολο να συντηρηθεί και το κυριότερο η επαλήθευση των χρονικών περιορισμών γίνεται πολύ δύσκολη. Συνέπεια αυτής της προσέγγισης είναι ότι το λογισμικό ελέγχου που παράγεται με την εμπειρική τεχνική μπορεί να έχει εξαιρετικά απρόβλεπτη συμπεριφορά. Το σύστημα θα μπορούσε να λειτουργεί καλά για μια μεγάλη χρονική περίοδο, αλλά να καταρρεύσει σε ορισμένες σπάνιες, αλλά υπαρκτές καταστάσεις. Οι συνέπειες της αστοχίας ενός συστήματος πραγματικού χρόνου μερικές φορές μπορεί να είναι πολύ καταστροφικές.

Ο χρόνος είναι το κύριο χαρακτηριστικό που διακρίνει τα συστήματα πραγματικού χρόνου από τα άλλα συστήματα. Ο όρος χρόνος δηλώνει ότι η ορθότητα του συστήματος δεν εξαρτάται μόνο από το αποτέλεσμα του υπολογισμού, αλλά και το χρονικό διάστημα κατά στο οποίο παράγονται τα αποτελέσματα. Ο όρος πραγματικός δείχνει ότι η αντίδραση των συστημάτων στα εξωτερικά γεγονότα πρέπει να συμβεί κατά τη διάρκεια της εξέλιξής τους. Κατά συνέπεια, η ώρα του συστήματος (εσωτερικός χρόνος) πρέπει να

μετράται με την ίδια χρονική κλίμακα που χρησιμοποιείται για τη μέτρηση του χρόνου στο ελεγχόμενο περιβάλλον (εξωτερικό ώρα).

Η έννοια του χρόνου δεν είναι ιδιότητα του συστήματος ελέγχου αλλά του περιβάλλοντος στο οποίο λειτουργεί το σύστημα ελέγχου [2]. Το σχήμα 2.1 απεικονίζει ένα γενικό σύστημα πραγματικού χρόνου που ελέγχει ένα φυσικό σύστημα.



Σχήμα 2.1 Γενικό σύστημα πραγματικού χρόνου.[2].

Μία άλλη λανθασμένη άποψη είναι ότι η πρόοδος στο υλικό των υπολογιστών θα καλύψει τις απαιτήσεις για πραγματικό χρόνο. Αν και η πρόοδος της τεχνολογίας θα βελτιώνει την απόδοση του συστήματος και θα αυξήσει την υπολογιστική ταχύτητα κατά πολύ αυτό δεν σημαίνει ότι οι χρονικές προθεσμίες θα ικανοποιούνται αυτόματα [2].

Στην πραγματικότητα, ο στόχος των γρήγορων υπολογιστών είναι να ελαχιστοποιήσουν την μέση χρονική απόκριση για ένα δεδομένο σύνολο εργασιών, ενώ ο στόχος των υπολογιστικών συστημάτων πραγματικού χρόνου είναι να ανταποκριθούν εμπρόθεσμα στις ιδιαίτερες χρονικές απαιτήσεις της κάθε εργασίας [1].

2.1.1 Κατηγορίες συστημάτων πραγματικού χρόνου.

Ένα σύστημα πραγματικού χρόνου αντί να είναι γρήγορο πρέπει να έχει προβλέψιμη χρονική συμπεριφορά. Για να επιτευχθεί η προβλεψιμότητα (predictability) πρέπει να χρησιμοποιηθούν νέες μεθοδολογίες σε κάθε στάδιο της ανάπτυξης μιας εφαρμογής πραγματικού χρόνου, από το σχεδιασμό μέχρι τη δοκιμή. Η κύρια διαφορά ανάμεσα σε ένα έργο πραγματικού χρόνου και ένα έργο μη πραγματικού χρόνου είναι ότι το έργο πραγματικού χρόνου, έχει μια *προθεσμία* (deadline), η οποία είναι ο μέγιστος χρόνος εντός του οποίου πρέπει να έχει ολοκληρωθεί η εκτέλεσή του.

Ανάλογα με τις συνέπειες που μπορεί να προκύψουν λόγω της εκπρόθεσμης ολοκλήρωσης ενός έργου πραγματικού χρόνου έχουμε τρείς κατηγορίες:

Αυστηρό (hard): η εκπρόθεσμη ολοκλήρωση ενός έργου έχει καταστροφικές συνέπειες για το σύστημα.

Σταθερό (firm): η εκπρόθεσμη ολοκλήρωση ενός έργου δίνει άχρηστα αποτελέσματα για το σύστημα, αλλά δεν προκαλεί καμία ζημιά.

Ήπιο (soft): η εκπρόθεσμη ολοκλήρωση ενός έργου δίνει αποτελέσματα με κάποια χρησιμότητα για το σύστημα αλλά προκαλεί υποβάθμιση των επιδόσεων του.

Ένα λειτουργικό σύστημα πραγματικού χρόνου που είναι σε θέση να χειριστεί έργα με αυστηρές προθεσμίες λέγεται αυστηρό σύστημα πραγματικού χρόνου. Συνήθως οι ε-

φαρμογές πραγματικού χρόνου περιλαμβάνουν έργα με αυστηρές, σταθερές αλλά και ήπιες προθεσμίες. Ένα αυστηρό σύστημα πραγματικού χρόνου θα πρέπει να σχεδιαστεί ώστε να μπορεί να χειριστεί όλες τις κατηγορίες χρησιμοποιώντας διαφορετικές στρατηγικές για κάθε μία.

Σε γενικές γραμμές, όταν μια εφαρμογή αποτελείται από ένα υβριδικό σύνολο έργων, όλες οι εργασίες με αυστηρές προθεσμίες θα πρέπει να είναι εγγυημένες πάντα, οι εργασίες με σταθερές προθεσμίες θα πρέπει να είναι εγγυημένες όταν είναι δυνατόν ενώ οι εργασίες με ήπιες προθεσμίες να εκτελούνται όσο το δυνατόν εντός των προθεσμιών τους.

2.1.2 Βασικές ιδιότητες συστημάτων πραγματικού χρόνου.

Μερικές βασικές ιδιότητες που πρέπει να έχουν τα συστήματα πραγματικού χρόνου για την υποστήριξη κρίσιμων χρονικά εφαρμογών είναι:

Εμπρόθεσμη ολοκλήρωση. Τα αποτελέσματα πρέπει να είναι σωστά και έγκαιρα.

Προβλεψιμότητα. Το σύστημα πρέπει να γνωρίζει τις συνέπειες της οποιασδήποτε απόφασης χρονοπρογραμματισμός (Scheduling). Στις κρίσιμες εφαρμογές, όλες οι χρονικές απαιτήσεις θα πρέπει να διασφαλίζονται πριν τεθεί το σύστημα σε λειτουργία. Εάν κάποια εργασία δεν μπορεί να ολοκληρωθεί εντός της προθεσμίας της το σύστημα πρέπει να το γνωρίζει εκ των προτέρων έτσι ώστε να μπορεί προγραμματίσει εναλλακτικές ενέργειες για να χειριστεί αυτή την κατάσταση.

Αποδοτικότητα. Τα περισσότερα συστήματα πραγματικού χρόνου είναι ενσωματωμένα σε μικρές συσκευές με αυστηρούς περιορισμούς χώρου, βάρους, καταναλισκόμενης ενέργειας, μνήμης και υπολογιστικής δύναμης. Στα συστήματα αυτά, η αποτελεσματική διαχείριση των διαθέσιμων πόρων από το λειτουργικό σύστημα είναι ουσιαστική για την επίτευξη της επιθυμητής απόδοσης.

Ευρωστία. Τα συστήματα πραγματικού χρόνου δεν πρέπει να καταρρεύσουν όταν φτάνουν σε συνθήκες φορτίου αιχμής. Η διαχείριση της υπερφόρτωσης και η προσαρμόσιμη συμπεριφορά είναι βασικά χαρακτηριστικά των συστημάτων πραγματικού χρόνου για να χειρίζονται τις μεταβλητές ανάγκες σε πόρους και τις μεγάλες εναλλαγές φορτίου.

Ανοχή σε σφάλματα. Αστοχίες του υλικού ή του λογισμικού δεν πρέπει να προκαλέσουν την ολική κατάρρευση του συστήματος

Συντηρησιμότητα. Η αρχιτεκτονική του συστήματος πραγματικού χρόνου θα πρέπει να έχει αρθρωτή δομή για να εξασφαλιστεί ότι θα είναι εύκολο να γίνει η συντήρηση και οι ενδεχόμενες τροποποιήσεις του συστήματος.

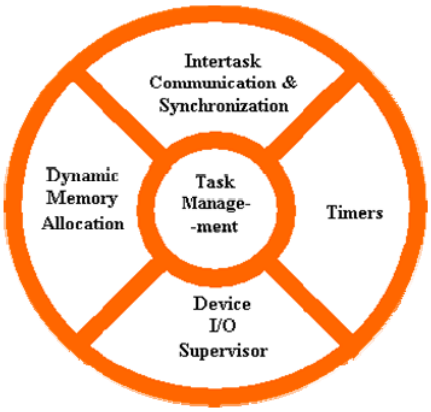
2.2 Βασικές έννοιες λειτουργικών συστημάτων πραγματικού χρόνου.

Σε αυτή την παράγραφο θα εστιάσουμε την προσοχή μας στις βασικές έννοιες των λειτουργικών συστημάτων με έμφαση στα πραγματικού χρόνου. Το τμήμα του λειτουργικού συστήματος που παρέχει τις πιο βασικές υπηρεσίες στην εφαρμογή που τρέχει σε ένα επεξεργαστή είναι ο *πυρήνας* (kernel).

2.2.1 Η δομή του πυρήνα.

Ο πυρήνας ενός RTOS παρέχει ένα επίπεδο αφαίρεσης που κρύβει από την εφαρμογή το υλικό και τα χαρακτηριστικά του ή τον επεξεργαστή στον οποίο θα τρέξει η εφαρμογή. Με βάση αυτή την αφαίρεση ο πυρήνας παρέχει πέντε βασικές κατηγορίες υπηρεσιών στην εφαρμογή που φαίνονται στο σχήμα 2.2 [3]

Η πιο βασική υπηρεσία βρίσκεται στο κέντρο και είναι η *διαχείριση εργασιών* (task management). Το σύνολο των υπηρεσιών που υποστηρίζει επιτρέπει στους προγραμματιστές να σχεδιάσουν το λογισμικό τους σαν ξεχωριστά κομμάτια που το καθένα χειρίζεται ένα ξεχωριστό τμήμα, ένα ξεχωριστό στόχο και πιθανό έχει τις δικές του χρονικές προθεσμίες.



Σχήμα 2.2 Η δομή ενός RTOS.

Κάθε ξεχωριστό τμήμα λογισμικού το λέμε εργασία (task). Οι υπηρεσίες σε αυτό το τμήμα περιλαμβάνουν την ικανότητα δημιουργίας και απόδοσης προτεραιότητας σε μία εργασία και τον χρονοπρογραμματισμό των εργασιών ενώ το σύστημα λειτουργεί. Ο χρονοπρογραμματιστής(scheduler) εργασιών ελέγχει την εκτέλεση των εργασιών του λογισμικού της εφαρμογής και μπορεί να τις κάνει να εκτελούνται με χρονικούς περιορισμούς.

Η δεύτερη κατηγορία υπηρεσιών του πυρήνα είναι η επικοινωνία μεταξύ των εργασιών. Αυτές οι υπηρεσίες κάνουν εφικτό το πέρασμα πληροφοριών μεταξύ των εργασιών χωρίς κίνδυνο καταστροφής τους και κάνουν εφικτό τον συντονισμό των εργασιών ώστε να συνεργάζονται παραγωγικά.

Επειδή πολλά συστήματα έχουν αυστηρές χρονικές απαιτήσεις τα περισσότερα RTOS έχουν βασικές υπηρεσίες χρονομετρητών (timer) όπως χρονικής καθυστέρησης εργασιών και λήξης χρόνου.

Πολλά, αλλά όχι όλα, παρέχουν υπηρεσίες δυναμικής κατανομή μνήμης. Αυτή η κατηγορία υπηρεσιών επιτρέπει στην εργασία να δανείζετε τμήματα μνήμης RAM για προσωρινή χρήση. Συχνά αυτά τα τμήματα μεταφέρονται από εργασία σε εργασία, με την έννοια της γρήγορης επικοινωνίας, μεταφέροντας μεγάλο όγκο δεδομένων από εργασία σε εργασία. Μερικά πολύ μικρά RTOS που στοχεύουν σε περιβάλλοντα με μικρή μνήμη δεν παρέχουν τέτοιες υπηρεσίες.

Πολλοί πυρήνες παρέχουν υπηρεσίες συντονισμού συσκευών εισόδου και εξόδου. Αυτές οι υπηρεσίες αν είναι διαθέσιμες παρέχουν ένα ενιαίο πλαίσιο εργασίας για πρόσβαση και οργάνωση των πολλών οδηγών συσκευών υλικού.

Πολλά RTOS παρέχουν επιπλέον ένα αριθμό πρόσθετων τμημάτων για υπηρεσίες υψηλού επιπέδου όπως είναι συστήματα αρχείων, δικτυακή επικοινωνία, διαχείριση δικτύων, διαχείριση βάσεων δεδομένων και γραφικά περιβάλλοντα επικοινωνίας. Καίτοι αυτά τα τμήματα είναι πολύ μεγαλύτερα και πολύ πιο πολύπλοκα από τον πυρήνα βασίζονται στην παρουσία του και παίρνουν ισχύ από τις βασικές υπηρεσίες που αυτός μπορεί να παρέχει. Κάθε ένα από αυτά τα τμήματα περιλαμβάνεται στο σύστημα μόνο αν οι υπηρεσίες του είναι αναγκαίες για την υλοποίηση της εφαρμογής.

Στη συνέχεια θα περιγράψουμε τις βασικές υπηρεσίες του πυρήνα για διαχείριση εργασιών, την επικοινωνία μεταξύ των εργασιών, τον συγχρονισμό και δυναμική κατανομή της μνήμης.

2.2.2 Βασικές υπηρεσίες πυρήνα.

Πολλά λειτουργικά συστήματα μη πραγματικού χρόνου παρέχουν παρόμοιες υπηρεσίας πυρήνα. Η βασική διαφορά μεταξύ των λειτουργικών συστημάτων γενικού σκοπού και των RTOS είναι η ανάγκη τα δεύτερα να είναι πλήρως ντετερμινιστικά. Ο όρος ντετερμινιστικά σημαίνει ότι οι υπηρεσίες χρειάζονται συγκεκριμένο, γνωστό από πριν, χρόνο να ολοκληρωθούν. Στη θεωρία αυτά μπορούν να αποδοθούν με μαθηματικούς τύπους. Αυτοί οι τύποι δεν περιλαμβάνουν τμήματα τυχαίου χρόνου. Τυχαία τμήματα σε χρόνο εξυπηρέτησης μπορεί να οδηγήσουν σε τυχαίες καθυστερήσεις τις εφαρμογές και κάποιες προθεσμίες να χαθούν. Αυτό είναι ένα μη αποδεκτό σενάριο για τα συστήματα πραγματικού χρόνου.

Τα γενικά λειτουργικά συστήματα συνήθως είναι μη ντετερμινιστικά. Οι υπηρεσίες τους μπορεί να εισάγουν τυχαίες καθυστερήσεις στις εφαρμογές και έτσι δημιουργούνται αυξομειώσεις στην απόκριση της εφαρμογής. Στα RTOS οι περισσότερες υπηρεσίες του πυρήνα παρέχουν σταθερό χρόνο εκτέλεσης ανεξάρτητα του φόρτου εργασιών.

2.2.3 Χρονοπρογραμματισμός εργασιών.

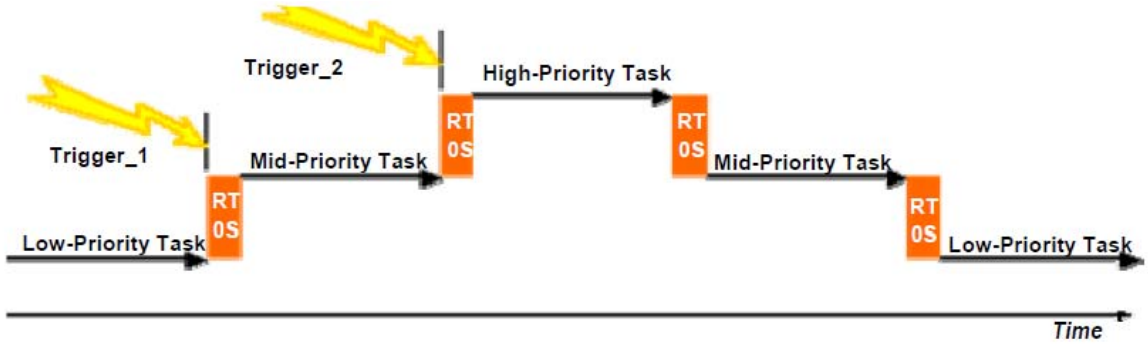
Τα περισσότερα RTOS κάνουν χρονοπρογραμματισμό με ένα σχήμα που λέγεται προεκτοπιστικός (preemptive) χρονοπρογραμματισμός βασισμένος σε προτεραιότητες. Κάθε εργασία μιας εφαρμογής θα πάρει μία προτεραιότητα. Οι εργασίες που έχουν υψηλότερη προτεραιότητα απαιτούν και γρηγορότερη απόκριση. Η γρήγορη απόκριση είναι εφικτή από την προεκτοπιστική φύση του χρονοπρογραμματιστή. Προεκτοπιστική σημαίνει ότι ο χρονοπρογραμματιστής μπορεί να διακόψει την εκτέλεση μιας εργασίας σε οποιοδήποτε σημείο της εκτέλεση της, αν έρθει για εκτέλεση μία εργασία με υψηλότερη προτεραιότητα.

Ο βασικός κανόνας είναι ότι σε κάθε χρονική στιγμή το υψηλότερης προτεραιότητας έργο που είναι έτοιμο για εκτέλεση θα είναι το έργο που θα εκτελεστεί. Μ’ άλλα λόγια αν δύο εργασίες, μία χαμηλής και μία υψηλής προτεραιότητας είναι έτοιμες για εκτέλεση, ο χρονοπρογραμματιστής θα επιλέξει την υψηλότερης προτεραιότητας να εκτελεστεί πρώτη. Η χαμηλής προτεραιότητας εργασία θα περιμένει.

Στην περίπτωση που μία εργασία υψηλής προτεραιότητας έγινε έτοιμη, αφού κάποια άλλη χαμηλότερης ξεκίνησε να εκτελείται, ένας προεκτοπιστικός αλγόριθμος με προτεραιότητες θα κάνει τα εξής: θα επιτρέψει, στη χαμηλής προτεραιότητας εργασία να ολοκληρώσει την τρέχουσα assembly εντολή (όχι όμως να ολοκληρώσει την εντολή της γλώσσας υψηλού επιπέδου), θα σταματήσει την εκτέλεση της εργασίας, θα αποθηκεύσει το περιβάλλον εκτέλεσης και θα αρχίσει την εκτέλεση της εργασίας με την υψηλότερη προτεραιότητα. Μετά την ολοκλήρωση της εργασίας υψηλού επιπέδου, θα φορτωθεί και πάλι η εργασία χαμηλού επιπέδου από το σημείο που διακόπηκε και θα συνεχίσει την εκτέλεσή της.

Στο σχήμα 2.3 φαίνεται σχηματικά η προεκτόπιση μίας εργασίας από μία άλλη υψηλότερης προτεραιότητας. Το έργο mid-Priority task εκτοπίζει το low-Priority task και αυτό με τη σειρά του εκτοπίζεται από το High-Priority task.

Κάθε φορά που ο χρονοπρογραμματιστής θα ενεργοποιείται είτε από ένα συμβάν λογισμικού είτε από ένα εξωτερικό γεγονός πρέπει να εκτελέσει τα παρακάτω βήματα:



Σχήμα 2.3 Προεκτοπισμός βάση προτεραιοτήτων.

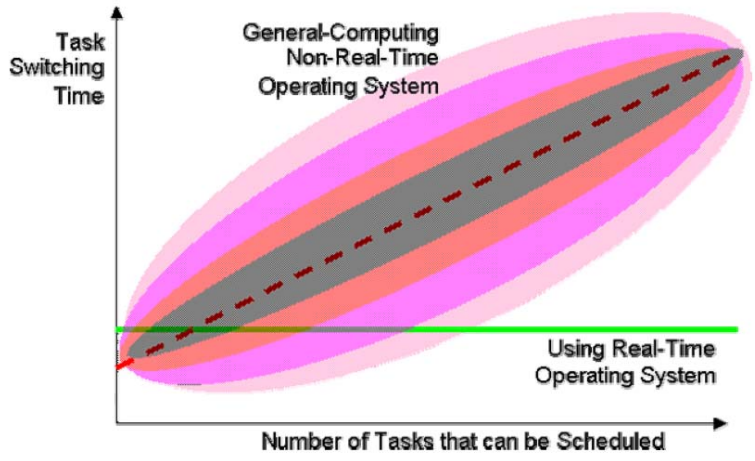
1. Να καθορίσει αν πρέπει να συνεχίσει να εκτελείται η τρέχουσα εργασία.
2. Αν όχι τότε
   - Να καθορίσει ποιο εργασία θα εκτελεστεί στη συνέχεια.
   - Να σώσει το περιβάλλον εκτέλεσης της τρέχουσας εργασίας για να την συνεχίσει αργότερα.
   - Να ορίσει το περιβάλλον της εργασίας που θα εκτελεστεί.
   - Να εκκινήσει την εκτέλεση της εργασίας.

Αυτά τα πέντε βήματα μαζί λέγονται εναλλαγή περιβάλλοντος εργασίας.

2.2.4 Εναλλαγή εργασιών (task switch).

Ο χρόνος που απαιτείται για την εναλλαγή μίας εργασίας είναι σημαντικός στην αξιολόγηση των RTOS.

Στο σχήμα 2.4 βλέπουμε ότι για ένα λειτουργικό σύστημα γενικού σκοπού ο χρόνος εναλλαγής αυξάνει καθώς αυξάνει ο αριθμός των έργων που χρονοπρογραμματίζονται. Ο ακριβής χρόνος δεν είναι η κόκκινη διακεκομμένη γραμμή αλλά για κάθε στιγμιότυπο μπορεί να είναι λίγο επάνω ή λίγο κάτω από αυτόν. Οι σκιασμένες περιοχές γύρω από την διακεκομμένη δείχνουν αυτή την τάση. Από την άλλη πλευρά η πράσινη συμπαγής γραμμή δείχνει τον χρόνο εναλλαγής που είναι χαρακτηριστικός των RTOS. Είναι σταθερός ανεξάρτητος του αριθμού των έργων.



Σχήμα 2.4 Χρόνος εναλλαγής έργου.

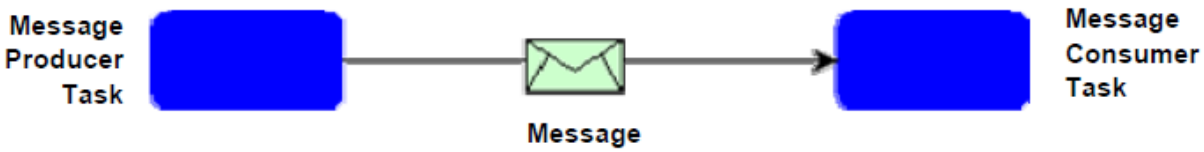
Σε μερικές περιπτώσεις, περιοχή κάτω αριστερά στο σχήμα, ο χρόνος εναλλαγής σε ένα γενικού σκοπού λειτουργικό σύστημα είναι μικρότερος από αυτόν του RTOS. Ο όρος

πραγματικός χρόνος δεν σημαίνει όσο πιο γρήγορα γίνεται, αλλά απαίτηση για σταθερή χρονική απόκριση, επαναληπτικότητα και χρονική απόδοση γνωστή από πριν. Καίτοι ένα γενικού σκοπού λειτουργικό σύστημα μπορεί να κάνει κάποιες εναλλαγές γρηγορότερα για μικρό αριθμό έργων μπορεί εξ ίσου να εισάγει μεγάλες χρονικές καθυστερήσεις όταν υπάρχει μεγάλος αριθμός εργασιών. Μετά τις 5 έως 10 εργασίες, τα RTOS αποκρίνονται πιο γρήγορα από τα γενικού σκοπού λειτουργικά συστήματα.

2.2.5 Συγχρονισμός και επικοινωνία μεταξύ διεργασιών.

Τα περισσότερα λειτουργικά συστήματα παρέχουν πολλούς μηχανισμός για επικοινωνία και συγχρονισμό διεργασιών. Αυτοί οι μηχανισμοί είναι αναγκαίοι σε ένα προεκτοπιστικό περιβάλλον με πολλές εργασίες, γιατί χωρίς αυτούς οι διεργασίες μπορούν ή να αλληλοπαρεμβάλλονται ή να επικοινωνούν με αλλοιωμένες πληροφορίες.

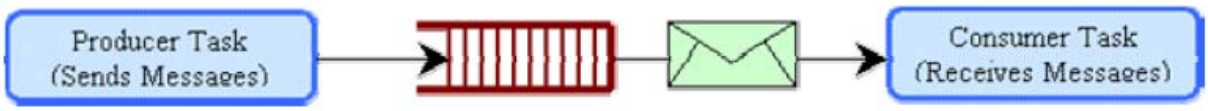
Για παράδειγμα μία εργασία μπορεί να προεκτοπιστεί όταν είναι στη μέση μίας ενημέρωσης ενός πίνακα δεδομένων. Αν η επόμενη εργασία, που την εκτόπισε, πρέπει να διαβάσει αυτά τα δεδομένα από τον πίνακα θα διαβάσει ένα συνδυασμό από κάποια ενημερωμένα και κάποια μη ενημερωμένα. Αυτός δεν είναι αποδεκτό. Για την αποφυγή τέτοιων λαθών παρέχεται ένας μηχανισμός για επικοινωνία και συγχρονισμό μεταξύ εργασιών. Τα περισσότερα έχουν μερικούς μηχανισμούς και καθένας τους είναι βελτιστοποιημένος για το αξιόπιστο πέρασμα ενός διαφορετικού είδους πληροφορίας από εργασία σε εργασία.



Σχήμα 2.5 Επικοινωνία διεργασιών μέσω μηνύματος.

Ίσως ο πιο δημοφιλής τρόπος επικοινωνίας μεταξύ εργασιών είναι το πέρασμα δεδομένων από τη μία εργασία στην άλλη. Τα περισσότερα RTOS έχουν ένα τέτοιο μηχανισμό όπως φαίνεται στο σχήμα 2.5. Κάθε μήνυμα μπορεί να περιέχει ένα πίνακα ή ένα απομονωτή(buffer) με δεδομένα.

Αν τα μηνύματα στέλνονται γρηγορότερα από ότι μπορούν να διαβαστούν, τα RTOS παρέχουν ουρές μηνυμάτων, που μπορούν να κρατήσουν τα μηνύματα μέχρι να έρθει η σειρά του να επεξεργαστούν. Αυτό απεικονίζεται στο σχήμα 2.6



Σχήμα 2.6 Επικοινωνία διεργασιών μέσω μηνύματος και ουράς.

Ένας άλλο είδος επικοινωνίας μεταξύ εργασιών είναι το πέρασμα πληροφορίας συγχρονισμού από τη μία εργασία στην άλλη. Τα περισσότερα RTOS παρέχουν *σηματοφορείς* (semaphore) ή μηχανισμούς αμοιβαίου αποκλεισμού (mutexes) για δέσμευση συγκεκριμένων πόρων για αποκλειστική χρήση. Οι πόροι αποδεσμεύονται όταν τελειώνει η χρήση τους. Άλλοι μηχανισμοί συγχρονισμού είναι τα σήματα, οι σημαίες συμβάντων και η τεχνική διαβίβασης μηνύματος όπως κάνουν για τα δεδομένα.

2.2.6 Ντετερμινισμός και γρήγορο πέρασμα μηνυμάτων.

Η επικοινωνία εργασιών μέσω μηνυμάτων είναι μια άλλη περιοχή όπου διαφορετικά συστήματα διαφέρουν σε χρονικά χαρακτηριστικά. Τα περισσότερα λειτουργικά συστήματα γράφουν δυο φορές τα μηνύματα καθώς τα μεταφέρουν μέσω ουράς από εργασία σε εργασία. Η πρώτη εγγραφή γίνεται από τον αποστολέα σε μία περιοχή της μνήμης στην οποία υλοποιείται η ουρά μηνυμάτων και η δεύτερη εγγραφή όταν αυτά αντιγράφονται απ' αυτή την περιοχή αποστολής, στην εργασία παραλήπτη. Αυτό δεν είναι χρονικά ντετερμινιστικό γιατί καθώς το μέγεθος του μηνύματος αυξάνει, ο χρόνος αυξάνει.

Μία προσέγγιση που αποφεύγει αυτή τη συμπεριφορά και επιταχύνει τη λειτουργία του συστήματος είναι η απόδοση από το λειτουργικό σύστημα ενός δείκτη (pointer) σ' αυτή την περιοχή και στη συνέχεια μεταβιβάζει αυτό το δείκτη στον παραλήπτη του μηνύματος χωρίς να μετακινηθεί το περιεχόμενο του μηνύματος. Με σκοπό την αποφυγή συγκρούσεων των προσπελάσεων το λειτουργικό σύστημα στη συνέχεια πρέπει να πάει στην εργασία που έστειλε τον δείκτη και να τον διαγράψει. Για μεγάλα μηνύματα αυτό ελαχιστοποιεί την ανάγκη για αντιγραφή και ελαχιστοποιεί τον μη ντετερμινισμό.

2.2.7 Δυναμική κατανομή μνήμης.

Ο ντετερμινισμός του χρόνου εξυπηρέτησης είναι ένα πρόβλημα στη δυναμική κατανομή μνήμης. Πολλά λειτουργικά συστήματα γενικού σκοπού παρέχουν υπηρεσίες δυναμικής κατανομής μνήμης από τον σωρό. Οι υπηρεσίες malloc και free της C εργάζονται με τον σωρό (heap). Οι εργασίες μπορούν να δανειστούν προσωρινά μερική μνήμη από το σωρό και να ορίσουν το μέγεθος της. Όταν η εργασία ολοκληρωθεί η μνήμη επιστρέφεται.

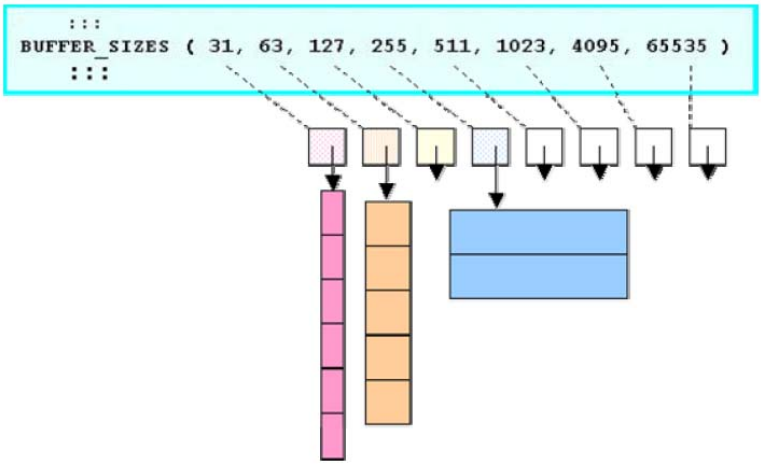
Ο σωρός υποφέρει από το φαινόμενο που ονομάζεται εξωτερικός κατακερματισμός μνήμης και μειώνει τις υπηρεσίες του σωρού. Αυτός προκαλείται από το γεγονός ότι ο buffer που επιστρέφεται στο σωρό και μπορεί να χρησιμοποιηθεί στο μέλλον σε επόμενη χρήση τεμαχίζεται σε ακόμη μικρότερα τμήματα. Μετά από μερικούς τέτοιους κύκλους, malloc και free, μπορεί να εμφανιστούν μικρά τμήματα μνήμης μεταξύ των buffers που είναι μικρά για να χρησιμοποιηθούν από κάποια εργασία αλλά όλα μαζί καταλαμβάνουν ένα μεγάλο μέρος του σωρού. Με την πάροδο του χρόνου ο σωρός μπορεί να έχει όλο και περισσότερα τέτοια κενά. Αυτό οδηγεί τελικά σε μια κατάσταση όπου θα ζητηθεί συγκεκριμένο μέγεθος μνήμης και θα απορριφθεί το αίτημα από το λειτουργικό σύστημα γιατί δεν θα υπάρχει ενιαίος χώρος τέτοιου μεγέθους έστω και αν συνολικά υπάρχει περισσότερος ελεύθερος χώρος στο σωρό.

Ο κατακερματισμός (fragmentation) μπορεί να λυθεί με λογισμικό που λέγεται αποκατακερματιστής. Ο αλγόριθμός αυτός δεν είναι ντετερμινιστικός βάζοντας καθυστερήσεις τυχαίας διάρκειας και εμφάνισης στις υπηρεσίες του σωρού. Αυτό συμβαίνει στα λειτουργικά συστήματα γενικού σκοπού.

Για τον προγραμματιστή εφαρμογών πραγματικού χρόνου που σκέφτεται να χρησιμοποιήσει ένα λειτουργικό σύστημα γενικού σκοπού για το σύστημά του αυτό τον βάζει στο δίλημμα: μπορεί το ενσωματωμένο σύστημα να μπει σε τυχαίας διάρκειας και εμφάνισης καθυστερήσεις όταν αρχίσει η διαδικασία του αποκατακερματισμού της μνήμης ή εναλλακτικά πρέπει το ενσωματωμένο σύστημα να επιτρέπει κατακερματισμό της μνήμης μέχρι την εμφάνιση αίτησης malloc στο σωρό που δεν θα ικανοποιηθεί καίτοι ο αθροιστικά ο συνολικός ελεύθερος χώρος θα ήταν αρκετός αν ήταν ενιαίος; Στα πραγματικού χρόνου συστήματα που είναι αναγκαίο να λειτουργούν για μεγάλα χρονικά διαστήματα με τον ίδιο τρόπο καμιά από αυτές τις τεχνικές δεν είναι αποδεκτή.

Τα RTOS αντί του σωρού παρέχουν άλλες τεχνικές δυναμικής κατανομής μνήμης που δεν παρουσιάζουν το πρόβλημα του κατακερματισμού. Το επιτυγχάνουν βάζοντας όρια

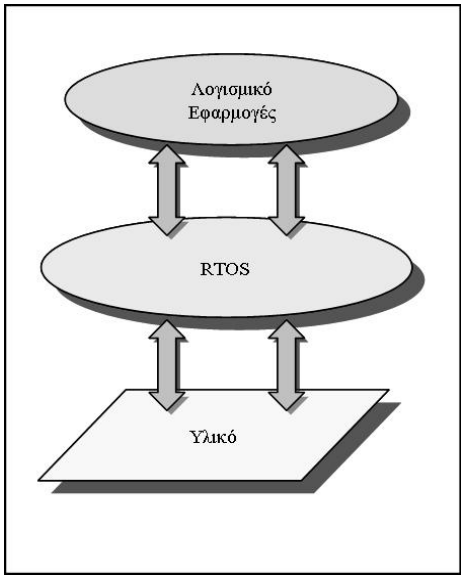
στην ποικιλία των τμημάτων μνήμης που είναι διαθέσιμα στις εφαρμογές. Παρόλο που αυτή η τεχνική είναι λιγότερο εύκαμπτη από το σωρό αποφεύγει το φαινόμενο του κατακερματισμού. Για παράδειγμα ο μηχανισμός pools για κατανομή μνήμης επιτρέπει την απόδοση μνήμης σε 4 ή 8 διαφορετικά μεγέθη. Αποφεύγει το φαινόμενο του κατακερματισμού μη επιτρέποντας τον τεμαχισμό σε μικρότερα τμήματα της μνήμης. Αυτός ο μηχανισμός φαίνεται στο σχήμα 2.7. Η κατανομή μνήμης με το μηχανισμό pool γίνεται σε ντετερμινιστικό χρόνο γνωστό από πριν.



Σχήμα 2.7 Ο μηχανισμός pool στην κατανομή της μνήμης.

2.2.8 Πυρήνες πραγματικού χρόνου.

Ο πυρήνας ενός RTOS παρέχει ένα επίπεδο αφαίρεσης που κρύβει από την εφαρμογή το υλικό και τα χαρακτηριστικά σχήμα 2.8. Στα ενσωματωμένα συστήματα είναι αρκετός μόνο ο πυρήνας. Στα πολύπλοκα συστήματα, όπως εκείνα που χρησιμοποιούνται για τον έλεγχο της εναέριας κυκλοφορίας, απαιτείται όλη η λειτουργικότητα ενός λειτουργικού συστήματος γενικού σκοπού και την παρέχουν κύρια τα εμπορικά RTOS.



Σχήμα 2.8 Ο πυρήνας.

Ο πυρήνας ασχολείται με τη διαχείριση πόρων συστήματος όπως ο επεξεργαστής, η μνήμη και ο χρόνος. Όλοι οι πόροι συστήματος διαμοιράζονται μεταξύ ανταγωνιζόμενων διεργασιών και αυτό πρέπει να διευθετηθεί με προσεκτικό τρόπο. Ο επεξεργαστής πρέπει να διαμοιραστεί, ώστε να αυξηθεί η απόδοση του και να εκτελεί τις εργασίες χω-

ρίς παραβίαση των προθεσμιών τους. Ο διαμοιρασμός πόρων μνήμης είναι απαραίτητος, επειδή οι πόροι είναι πεπερασμένοι.

Τα λειτουργικά συστήματα πραγματικού χρόνου οφείλουν να παρέχουν τρεις ειδικές λειτουργίες αναφορικά με τις διεργασίες: χρονοπρογραμματισμό, αποστολή, επικοινωνία και συγχρονισμό. Ο πυρήνας του λειτουργικού συστήματος είναι το μικρότερο δυνατό τμήμα που μπορεί να παρέχει αυτές τις λειτουργίες. Ο χρονοπρογραμματιστής καθορίζει ποια είναι η επόμενη διεργασία που θα εκτελεστεί, ενώ ο αποστολέας (dispatcher) είναι αυτός που κάνει τις απαραίτητες διαδικασίες, ώστε να ξεκινήσει μία διεργασία. Η επικοινωνία μεταξύ διεργασιών και ο συγχρονισμός διασφαλίζουν ότι οι διεργασίες συνεργάζονται.

Πίνακας 2.1.. Από το «γυμνό» υλικό στο λειτουργικό σύστημα.

Κατηγορία	**Λειτουργίες**
Λειτουργικό σύστημα	Κέλυφος διεπαφής χρήστη
Εκτελέσιμος πυρήνας	Υποστήριξη συστήματος αρχείων
Πυρήνας	Επικοινωνία και συγχρονισμός διεργασιών
Μικροπυρήνας	Χρονοπρογραμματισμός διεργασιών
Νανοπυρήνας	Διαχείριση νημάτων

Στον πίνακα 2-1 έχουμε τα διάφορα επίπεδα ενός λειτουργικού συστήματος και τις αντίστοιχες λειτουργίες που αυτό παρέχει. Προχωρώντας από κάτω προς τα πάνω, από το νανοπυρήνα προς το πλήρες λειτουργικό σύστημα, φαίνεται η επιπλέον παρεχόμενη λειτουργικότητα. Ένας νανοπυρήνας (nanokernel) παρέχει διαχείριση απλών νημάτων. ένας μικροπυρήνας (microkernel) παρέχει επιπλέον τον χρονοπρογραμματισμό διεργασιών. Ένας πυρήνας παρέχει συγχρονισμό και επικοινωνία μεταξύ των διεργασιών μέσω σηματοφορέων, γραμματοθυρίδων και άλλων μεθόδων. Ένας εκτελέσιμος πυρήνας πραγματικού χρόνου είναι ένας πυρήνας που περιλαμβάνει δέσμευση τμημάτων μνήμης, υπηρεσίες εισόδου-εξόδου και άλλα πολύπλοκα χαρακτηριστικά. Οι περισσότεροι εμπορικοί πυρήνες πραγματικού χρόνου είναι εκτελέσιμοι. Τέλος, το λειτουργικό σύστημα περιλαμβάνει τον πυρήνα αλλά παρέχει επιπλέον μία προηγμένη γενική διεπαφή χρήστη, ασφάλεια, ένα σύστημα διαχείρισης αρχείων και τμήματα για υποστήριξη εξειδικευμένων εργασιών.

2.3 Χαρακτηριστικά των λειτουργικών συστημάτων πραγματικού χρόνου.

Το πρότυπο IEEE POSIX 1003.1b[4] παρέχει μία λίστα από βασικές υπηρεσίες που πρέπει να υποστηρίζουν τα RTOS και περιγράφονται αναλυτικά στην παράγραφο 4.1.1. Επιπλέον άλλα βασικά χαρακτηριστικά που περιγράφονται σε μία πρόσφατη έρευνα για τα RTOS [5] είναι τα ακόλουθα:

Χαμηλή επιβάρυνση. Η μεταγωγή περιβάλλοντος για νήματα πρέπει να προσθέτει μικρή επιβάρυνση στο λειτουργικό σύστημα.

Προεκτόπιση. Το RTOS πρέπει να μπορεί να προεκτοπίσει το εκτελούμενο νήμα και να δώσει τον επεξεργαστή σε ένα άλλο με μεγαλύτερη προτεραιότητα. Πάλι αυτό το χαρακτηριστικό χρειάζεται για να επιτρέψει σε επείγοντα γεγονότα (πχ υπέρβαση ενός κρίσιμου ορίου) να διακόψουν μια εργασία χαμηλότερης προτεραιότητας.

Ντετερμινιστικός συγχρονισμός. Η ικανότητα πολλά νήματα να επικοινωνούν μεταξύ τους σε ορισμένο προβλέψιμο χρόνο.

Επίπεδα προτεραιότητας. Το RTOS πρέπει να παρέχει αρκετά επίπεδα προτεραιότητας για να επιτρέπει την αποδοτική υλοποίηση των εφαρμογών. Αυτό το χαρακτηριστικό είναι σημαντικό για την ενεργοποίηση του προεκτοπιστικού χρονοπρογραμματισμού με προτεραιότητες.

Προκαθορισμένες χρονικές καθυστερήσεις. Ο χρόνος της κλήσης σε APIs πρέπει να παρέχει αναμενόμενες χρονικές καθυστερήσεις. Αυτό το χαρακτηριστικό είναι χρήσιμο όταν βήματα της διεργασίας χρειάζεται να ξεκινήσουν σε συγκεκριμένο χρονικό διάστημα μετά την εκτέλεση του προηγούμενου βήματος.

Τα χαρακτηριστικά ενός RTOS είναι αναγκαία αλλά όχι ικανά για την υλοποίηση ενός συστήματος πραγματικού χρόνου. Άσχετα από το αν το RTOS παρέχει ένα αναγκαίο χαρακτηριστικό για το σύστημα αυτό είναι άνευ αξίας αν το υποκείμενο υλικό δεν παρέχει την απαραίτητη ισχύ. Η ταχύτητα του επεξεργαστή, η ταχύτητα πρόσβασης της μνήμης, ο χρόνος πρόσβασης των συσκευών επηρεάζουν τον παράγοντα της ταχύτητας για το υλικό. Χρειάζεται από το στάδιο του σχεδιασμού, επιβεβαίωση ότι το υλικό είναι ικανό να υποστηρίξει τους αυστηρούς χρονικούς περιορισμούς και να εγγυηθεί τον απαιτούμενο χειρότερο χρόνο εκτέλεσης μίας εργασίας στο πλήρους φορτίου σύστημα πραγματικού χρόνου ανεξάρτητα από το RTOS που θα χρησιμοποιηθεί. Αφού το υλικό αποδειχθεί κατάλληλο για την εφαρμογή πραγματικού χρόνου μετά μπορούμε να αξιολογήσουμε τα χαρακτηριστικά των RTOS.

2.4 Αξιολόγηση ενός RTOS.

Είναι καλό και ωφέλιμο να γνωρίζει κάποιος πώς να διαλέξει το κατάλληλο RTOS για μία εφαρμογή πραγματικού χρόνου. Η πρώτη ερώτηση που πρέπει να απαντηθεί είναι αν μια έτοιμη λύση είναι συμβατή με το λογισμικό που χρειάζεται να φιλοξενήσει. Πολλά εμπορικά RTOS έχουν τέτοιες λίστες.

Μία εφαρμογή πραγματικού χρόνου για ένα ενσωματωμένο σύστημα απαιτεί ένα RTOS με μικρό αποτύπωμα (footprint) και μικρή επιβάρυνση. Μία εφαρμογή πραγματικού χρόνου που τρέχει σε ένα PC μπορεί να υποστηρίξει πιο σύνθετα RTOS που παρέχουν πρόσθετα χαρακτηριστικά και πολύπλοκες διεπαφές χρήστη. Και οι δύο επιλογές λογισμικού και υλικού πρέπει να θεωρούνται μαζί όταν σχεδιάζουμε εφαρμογές πραγματικού χρόνου. Το σύστημα κοστίζει, ο χρόνος ανάπτυξης και οι πιθανότητες επιτυχίας εξαρτώνται από τα επιλογές που θα γίνουν στο στάδιο της σχεδίασης. Ένα από τα πιο σημαντικά κριτήρια, συχνά υπερτονισμένο, όταν αξιολογούμε ένα RTOS είναι η ικανότητα υποστήριξης εργαλείων όπως εκσφαλματωτής πραγματικού χρόνου και υποστήριξη κατάλληλων γλωσσών προγραμματισμού. Η χρήση πολύπλοκων ολοκληρωμένων περιβαλλόντων ανάπτυξης εφαρμογών είναι σήμερα κοινός τόπος. Τα περιβάλλοντα παρέχουν στους προγραμματιστές εργαλεία αλλά και παραδείγματα για τη γρήγορη ανάπτυξη λογισμικού, τις δοκιμές και τη διαχείριση των διαφόρων εκδόσεων. Όταν οι προγραμματιστές είναι εξοικειωμένοι με το περιβάλλον η παραγωγικότητά τους αυξάνει και η ικανότητα για εργασία σε ομάδα σ' ένα ομαδικό περιβάλλον βελτιώνεται.

Πρέπει η μηχανή που φιλοξενεί την ανάπτυξη να είναι ίδια με αυτή που θα τρέξει η εφαρμογή; Όχι πάντοτε. Μερικά περιβάλλοντα επιτρέπουν στους χρήστες να αναπτύσσουν την εφαρμογή σε μία πλατφόρμα και να κάνουν την εγκατάσταση σε άλλη. Προφανώς η ανάπτυξη σε ίδια πλατφόρμα βοηθά στο γρήγορο εντοπισμό των σφαλμάτων. Από την άλλη η ανάπτυξη σε μεγαλύτερες πλατφόρμες θα επιτρέψει τη χρήση πιο πολύπλοκων εργαλείων.

Εάν πρόκειται να γραφτεί τμήμα από τον κώδικα της εφαρμογής πραγματικού χρόνου προκύπτει το ερώτημα: Ποια προγραμματιστική γλώσσα είναι η πιο κατάλληλη; Η επι-

λογή της γλώσσας εξαρτάται από τις ικανότητες του προγραμματιστή και από το αν η γλώσσα υποστηρίζεται από το περιβάλλον ανάπτυξης που έχουμε. Η εκπαίδευση είναι απαραίτητη όταν χρησιμοποιείται νέα γλώσσα, αλλά αυτή η επιβάρυνση μπορεί να αξίζει εάν αυτή η γλώσσα είναι τμήμα ενός πλαισίου εργασίας που διευκολύνει την ανάπτυξη και παρέχει χαρακτηριστικά που είναι απαραίτητα σε μία εφαρμογή πραγματικού χρόνου.

Υπάρχουν πολλά μη τεχνικά κριτήρια που πρέπει επίσης να ληφθούν υπόψη. Η επιλογή υπαγορεύεται και από το κόστος. Για παράδειγμα μπορεί να επιλέξεις είτε ένα εμπορικό RTOS είτε να επιλέξεις ένα από την κοινότητα του ανοικτού λογισμικού. Σε εξαιρετικές περιπτώσεις (όπως για μικρή παραγωγή αλλά υψηλής αξιοπιστίας προϊόντα) μπορεί να απαιτηθεί η ανάπτυξη ενός RTOS από το μηδέν. Σε κάθε άλλη περίπτωση πρέπει να υπολογιστεί ποιο θα είναι το συνολικό κόστος. Εμπορικά RTOS μπορεί να έχουν υψηλό αρχικό κόστος κτήσης και να χρεώνουν δικαιώματα για κάθε πώληση του συστήματος. Η απόκτηση και πηγαίου κώδικα, για να γίνουν προσθήκες ή τροποποιήσεις, θα αυξήσει επιπλέον το κόστος. Το ανοικτό λογισμικό παρέχει RTOS με δωρεάν πρόσβαση στον πηγαίο κώδικα και στον δυαδικό κώδικα χωρίς δικαιώματα επί των πωλήσεων, αλλά συχνά απαιτείται και η αγορά υποστήριξης και μακροχρόνιας συμφωνίας με την κοινότητα των χρηστών. RTOS ανοιχτού κώδικα μπορεί μακροχρόνια να οδηγήσουν σε υψηλότερο κόστος λόγω χαμηλής αποδοτικότητας εργαλείων και έλλειψη τεχνικών υψηλής κατάρτισης.

Περιληπτικά το συνολικό κόστος εξαρτάται από το αρχικό κόστος κτήσης και από δικαιώματα χρήσεις, τα δικαιώματα επί των πωλήσεων, την υποστήριξη, την συντήρηση, την εκπαίδευση και τις συμβουλευτικές υπηρεσίες. Στην εργασία [6] υπάρχει μία εκτενής αναφορά για το ταίριασμα χαρακτηριστικών RTOS και ειδικών κριτηρίων για έργα. Σε κάθε περίπτωση η υλοποίηση των POSIX χαρακτηριστικών που παρέχονται στα χαρακτηριστικά ενός RTOS είναι ένα καλό σημείο εκκίνησης.

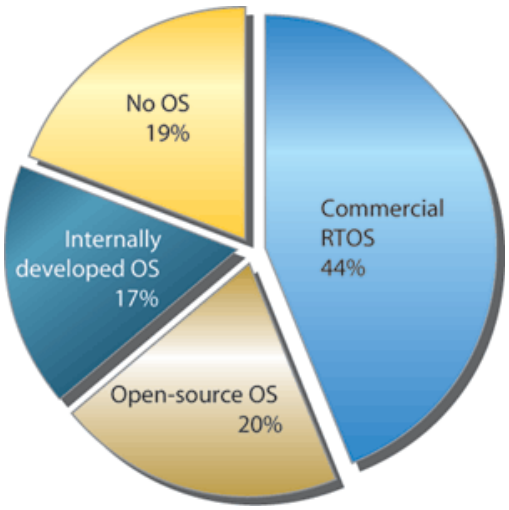
Με τέτοια πληθώρα συστημάτων και χαρακτηριστικών είναι δύσκολο να αποφασίσει κάποιος ποιό είναι το καλύτερο για την εφαρμογή του. Σε πρώτη θέση βρίσκετε η επιλογή του επεξεργαστή, η απόδοση πραγματικού χρόνου και το οικονομικό κόστος. Αυτοί οι παράγοντες δεν μπορούν να αλλάξουν αργότερα και βοηθούν στον περιορισμό των πιθανών επιλογών σε μικρότερο πλήθος προϊόντων.

Η καλύτερη αιτιολογία για την επιλογή ενός εμπορικού λειτουργικού συστήματος είναι το πλεονέκτημα του ότι το προϊόν είναι καλύτερα δοκιμασμένο και έτσι ποιο αξιόπιστο από ένα πυρήνα που έχει αναπτύξει κάποιος μόνος του. *Ένα από τα σημαντικότερα πράγματα που αγοράζουμε μαζί με το λειτουργικό σύστημα από ένα προμηθευτή είναι η εμπειρία του και η αξιοπιστία του προϊόντος του.*

2.5 Η αγορά των RTOS.

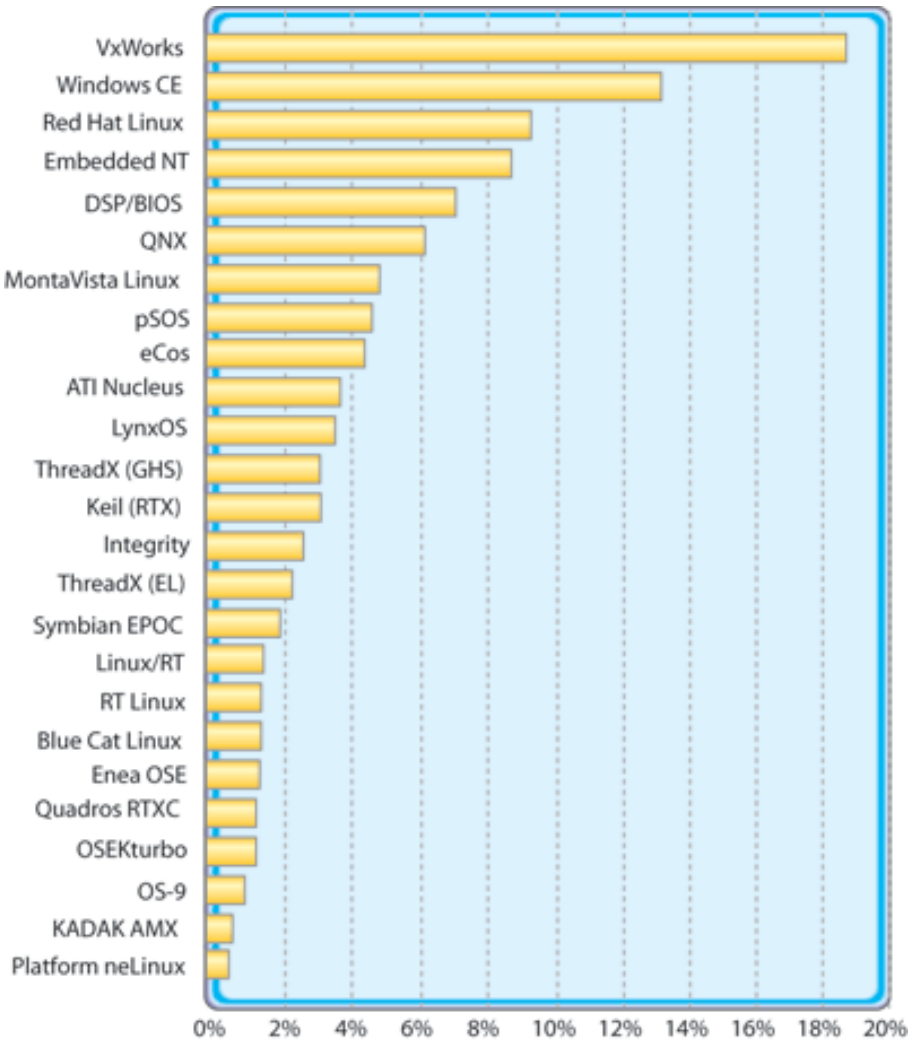
Σύμφωνα με μία εκτεταμένη έρευνα που έκανε και δημοσιεύει το περιοδικό EE Times (http://www.eetimes.com/discussion/other/4025539/Embedded-systems-survey-Operating-systems-up-for-grabs) ένας σημαντικός παράγοντας που επηρεάζει την επιλογή του λειτουργικού συστήματος είναι οι απαιτήσεις των πελατών και οι κανονιστικές διατάξεις. Τα μερίδια της αγοράς ανά κατηγορία απεικονίζονται στο σχήμα 2-9.

Η διείσδυση των εμπορικών RTOS είναι πολύ σημαντική στα συστήματα που χρησιμοποιούν υλικό 64bit και 32bit. Στα συστήματα 16bit υπάρχει ισορροπία και στα συστήματα 8bit σε πολύ μεγάλο ποσοστό ή δεν υπάρχει καθόλου λειτουργικό σύστημα ή είναι πυρήνες ανεπτυγμένοι in-house.

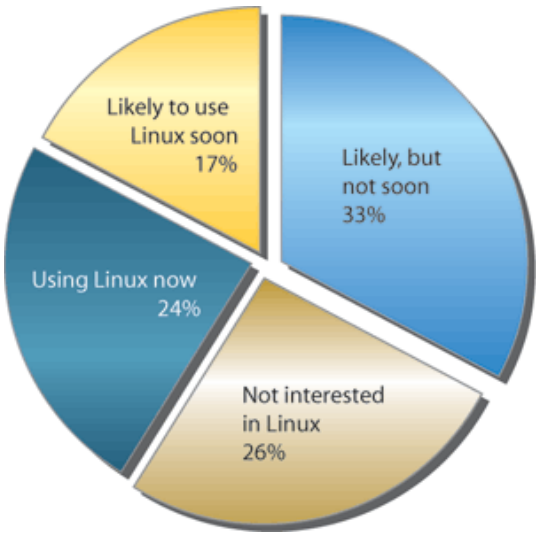


Σχήμα 2.9 Τα μερίδια της αγοράς.

Τα μερίδια των εμπορικών RTOS φαίνονται στο σχήμα 2.10. Αν λάβουμε υπόψη μας ότι υπάρχει και το τμήμα του ανοικτού λογισμικού όπου κυριαρχεί με 20% περίπου το linux σε διάφορες βλέπουμε ότι ο χώρος των RTOS είναι πολυκατακερματισμένος.



Σχήμα 2.10 Τα μερίδια των RTOS.



Σχήμα 2.11 Ενδιαφέρον για το linux.

Οι παράγοντες που κύρια επηρεάζουν την επιλογή ενός RTOS, κατά την έρευνα, είναι το συνολικό κόστος, τα δικαιώματα και τα χαρακτηριστικά πραγματικού χρόνου.

Το Linux παρόλο που δεν είναι εγγενώς RTOS έχει καταφέρει να αποκτήσει ένα σημαντικό κομμάτι της αγοράς. Η τάση του δείχνει να είναι αυξητική όχι τόσο από το γεγονός ότι είναι ανοικτό λογισμικό όσο από το ότι δεν υπάρχουν δικαιώματα στα παραγόμενα προϊόντα. Στο σχήμα 2.11 βλέπουμε τα συμπεράσματα της έρευνας για το ενδιαφέρον για μελλοντική χρήση κάποιας έκδοσης του Linux. Να σημειώσουμε ότι ένα μεγάλο ποσοστό από αυτούς που δήλωσαν ότι δεν ενδιαφέρονται για το linux οφείλεται στο ότι δεν είναι συμβατό με υπάρχοντα κώδικα που χρησιμοποιούν.

3. Χρονοπρογραμματισμός πραγματικού χρόνου.

Σε αυτό το κεφάλαιο θα περιγράψουμε τις βασικές έννοιες του χρονοπρογραμματισμού πραγματικού χρόνου, τους αλγορίθμους χρονοπρογραμματισμού περιοδικών έργων και θα παρουσιάσουμε προσομοιωτές χρονοπρογραμματισμού πραγματικού χρόνου. Η οργάνωση αυτού του κεφαλαίου είναι η ακόλουθη: Στην πρώτη παράγραφο αναφέρουμε τις βασικές έννοιες και τις παραμέτρους του χρονοπρογραμματισμού πραγματικού χρόνου. Στη δεύτερη παράγραφο περιγράφουμε τέσσερεις βασικούς αλγορίθμους. Στην τρίτη παρουσιάζουμε προσομοιωτές χρονοπρογραμματισμού πραγματικού χρόνου και τέλος στην τέταρτη παράγραφο δίνουμε παραδείγματα από τη χρήση του προσομοιωτή Cheddar.

3.1 Βασικές έννοιες.

Οι διεργασίες πραγματικού χρόνου είναι οι βασικές οντότητες που χρονοπρογραμματίζονται. Οι διεργασίες μπορεί να είναι περιοδικές, σποραδικές ή απεριοδικές. Σε αυτό το κεφάλαιο θα ασχοληθούμε με περιοδικές ανεξάρτητες διεργασίες. Μία διεργασία μπορεί να έχει αυστηρούς ή ήπιους περιορισμούς. Για τη μελέτη του χρονοπρογραμματισμού χρησιμοποιούμε ένα μοντέλο διεργασίας που προσδιορίζεται με χρονικά χαρακτηριστικά που περιλαμβάνουν βασικές και δυναμικές παραμέτρους.

3.1.1 Βασικές παράμετροι διεργασιών.

Οι βασικές παράμετροι παριστάνονται σχηματικά στο σχήμα 3.1 και είναι:

- r χρόνος άφιξης της διεργασίας (task release time) που προσδιορίζεται από τη χρονική στιγμή που γίνεται αίτηση για εκτέλεση της.
- C χρόνος εκτέλεσης της διεργασίας – στη χειρότερη περίπτωση - όταν ο επεξεργαστής εξυπηρετεί μόνον αυτή.
- D σχετική χρονική προθεσμία (task relative deadline) εκτέλεσης της διεργασίας που προσδιορίζει τη μέγιστη επιτρεπτή καθυστέρηση ολοκλήρωσής της.
- T περίοδος της διεργασίας (για περιοδικές διεργασίες).

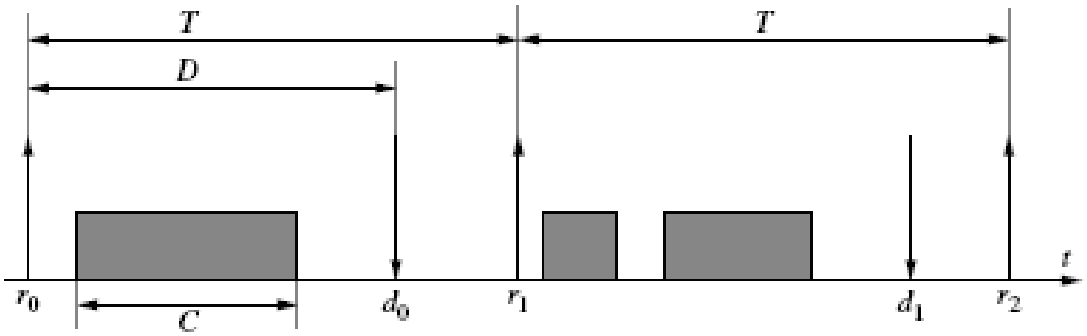
Όταν η διεργασία είναι αυστηρού πραγματικού χρόνου τότε η σχετική χρονική προθεσμία ολοκλήρωσής της μετατρέπεται σε απόλυτη χρονική προθεσμία d=r+D. Υπέρβαση των χρονικών ορίων της απόλυτης προθεσμίας, έχει αποτέλεσμα την αποτυχία τήρησης των χρονικών περιορισμών.

Μία διεργασία γράφεται ως τ(r0, c, d, t) όπου:

- r0 : ο χρόνος άφιξης της πρώτης αίτησης για εκτέλεση.
- c : ο χρόνος εκτέλεσης της διεργασίας.
- d : η σχετική προθεσμία.
- t : η περίοδος.
- rk : χρόνος άφιξης του (k+1) αιτήματος για εκτέλεση ↑.
- dk. : απόλυτη χρονική προθεσμία του (k+1) αιτήματος ↓.

Μία περιοδική διεργασία περιγράφεται από όλους τους παραπάνω χρονικούς παράγοντες. Κάθε φορά που μία διεργασία είναι σε κατάσταση ετοιμότητας, απευθύνει μια αίτηση εκτέλεσης. Οι διαδοχικοί χρόνοι που απευθύνει αυτήν την αίτηση είναι $r_k = r_0 + kT$, όπου r_0 είναι ο χρόνος της πρώτης άφιξης. Οι διαδοχικοί απόλυτοι χρόνοι θα είναι d_k =

r_k +D. Αν ισχύει D=T, η περιοδική διεργασία έχει σχετική χρονική προθεσμία εκτέλεσης ίση με την περίοδό της. Μια διεργασία είναι καλά δομημένη εάν ισχύει $0 \leq C \leq D \leq T$.



Σχήμα 3.1 Μοντέλο διεργασίας.

Η ποιότητα του χρονοπρογραμματισμού εξαρτάται από την ακρίβεια των παραπάνω παραμέτρων και έτσι ο λεπτομερής καθορισμός τους είναι πολύ σημαντικό ζήτημα για το σχεδιασμό ενός συστήματος πραγματικού χρόνου.

3.1.2 Δυναμικές παράμετροι διεργασιών.

Παρακάτω αναφέρονται κάποιες δυναμικές παράμετροι που βοηθούν στην παρακολούθηση της εκτέλεσης μιας διεργασίας:

s, χρόνος έναρξης της εκτέλεσης της διεργασίας.

e, χρόνος πέρατος της εκτέλεσης της διεργασίας.

D(t) = d-t, χρόνος που υπολείπεται από το χρονικό σημείο (t) που βρίσκεται η διεργασία μέχρι το χρόνο πέρατος της σχετικής της προθεσμίας. Ισχύει: $0 \leq D(t) \leq D$.

C(t), χρόνος που απομένει από το χρονικό σημείο που βρίσκεται η διεργασία μέχρι το χρονικό σημείο περάτωσής της. Ισχύει: $0 \leq C(t) \leq C$.

L = D – C, ονομαστική χαλαρότητα (nominal laxity), προσδιορίζει το χρονικό περιθώριο που έχει η διεργασία για να ολοκληρώσει τη λειτουργία της εντός των ορίων της προθεσμίας της.

L(t) = D(t) – C(t), σχετική χαλαρότητα (relative laxity), προσδιορίζει το χρονικό περιθώριο που απομένει στη υπό εξέλιξη διεργασία από το σημείο που βρίσκεται μέχρι να ολοκληρώσει τη λειτουργία της εντός της προθεσμίας της.

TR = e-r, χρόνος αντίδρασης της διεργασίας. Με σωστό χρονοπρογραμματισμό πρέπει $C \leq TR \leq D$.

CH(t) = C(t)/D(t), συντελεστής φόρτου σε μια υπό εξέλιξη διεργασία. Ισχύει $0 \leq CH(t) \leq C/T$ (εξ ορισμού αν e=d, CH(e)=0).

3.1.3 Καταστάσεις Διεργασιών.

Οι καταστάσεις στις οποίες μπορεί να βρεθεί μια διεργασία είναι οι ακόλουθες:

Επιλεγμένη (elected): η διεργασία εκτελείται. Ο επεξεργαστής έχει αναλάβει αποκλειστικά τη διεργασία αυτή τη στιγμή. Οι χρόνοι C(t) και D(t) μειώνονται συνεχώς.

Μπλοκαρισμένη (blocked): η διεργασία είναι μπλοκαρισμένη μέχρι να λάβει χώρα κάποιο εξωτερικό συμβάν (αποδέσμευση πόρου, σήμα συγχρονισμού, κλπ.). Οι χρόνοι L(t) και D(t) μειώνονται συνεχώς.

Σε ετοιμότητα (ready): η διεργασία είναι έτοιμη για εκτέλεση. Οι χρόνοι L(t) και D(t) μειώνονται συνεχώς.

Εκτός των χρονικών παραμέτρων, οι διεργασίες προσδιορίζονται και από άλλα χαρακτηριστικά όπως τα παρακάτω :

Προεκτοπίσιμες ή μη προεκτοπίσιμες: διεργασίες που βρίσκονται σε κατάσταση επιλεγμένη, και δεν θα μεταπέσουν σε άλλη κατάσταση πριν ολοκληρωθούν καλούνται μη προεκτοπίσιμες διεργασίες. Σε αντίθετη περίπτωση οι διεργασίες που βρίσκονται σε κατάσταση επιλεγμένη και διακόπτεται η εκτέλεσή τους για να μεταβούν στην κατάσταση ετοιμότητας, καλούνται προεκτοπίσιμες διεργασίες. Στην κατάσταση επιλεγμένη επιλέγεται να μεταβεί μια άλλη διεργασία.

Αλληλοεξάρτηση διεργασιών (dependency of tasks): οι διεργασίες μπορεί να αλληλεπιδρούν σύμφωνα με ένα κοινά συμφωνημένο τρόπο ή με την εκπομπή ενός μηνύματος ή τέλος με μια σαφή διαδικασία συγχρονισμού. Αυτό προϋποθέτει μια προγενέστερη σχέση μεταξύ των διεργασιών, η οποία θα είναι γνωστή πριν την εκτέλεσή τους και θα υποδεικνύει την προτεραιότητα εκτέλεσής τους. Έτσι, θα επιτυγχάνεται ευκολότερα ο αμοιβαίος αποκλεισμός δηλαδή η εξασφάλιση ότι μια διεργασία και μόνο αυτή χρησιμοποιεί την κρίσιμη περιοχή ενός πόρου ή ενός αρχείου ή μιας βάσης δεδομένων. Ο αμοιβαίος αποκλεισμός διαδραματίζει σημαντικό ρόλο καθώς αποτρέπει τα σφάλματα που συμβαίνουν όταν δύο ή περισσότερες διεργασίες έχουν ταυτόχρονη προσπέλαση σε μια κρίσιμη περιοχή. Υπάρχουν βέβαια και ανεξάρτητες διεργασίες που δεν διαμοιράζονται κρίσιμες περιοχές και έτσι ο χειρισμός τους όσο αφορά στον χρονοπρογραμματισμό είναι διαφορετικός.

Μέγιστο χρονικό περιθώριο (maximum jitter): ο χρόνος που μεσολαβεί από τη στιγμή της αίτησης μιας διεργασίας μέχρι τη στιγμή της έναρξης της εκτέλεσής της πρέπει να είναι γνωστός και να κυμαίνεται σε συνήθη όρια. Ο μέγιστος αυτός χρόνος μεταξύ των διεργασιών καλείται maximum jitter.

Βαθμός επείγοντος (urgency): οι χρονικές προθεσμίες της κάθε διεργασίας προσδιορίζουν το βαθμό του επείγοντος όσο αφορά στην παροχή των δεδομένων των διεργασιών. Σε δύο διεργασίες με ισοδύναμο βαθμός επείγοντος παρέχεται η ίδια χρονική προθεσμία.

3.1.4 Αλγόριθμοι Χρονοπρογραμματισμού.

Σε ένα σύστημα πραγματικού χρόνου, οι διεργασίες διέπονται από χρονικούς περιορισμούς και η εκτέλεσή τους είναι απαραίτητο να πραγματοποιείται μέσα σε σαφή χρονικά όρια. Αντικειμενικός σκοπός του χρονοπρογραμματισμού είναι να επιτρέψει στις διεργασίες να εκτελεστούν ικανοποιώντας αυτούς τους χρονικούς περιορισμούς. Ο τρόπος χρονοπρογραμματισμού πρέπει να προβλέπει ότι όλοι οι χρονικοί περιορισμοί θα εκπληρωθούν, όταν η εφαρμογή διεκπεραιώνεται με κανονικό τρόπο. Αν κάτι συμβεί στην υπό έλεγχο διαδικασία, θα πρέπει να υπάρχει η δυνατότητα να ενεργοποιηθούν διεργασίες που θα σημάνουν συναγερμό και τότε σκοπός του χρονοπρογραμματισμού είναι να κρατηθεί η όλη διαδικασία ασφαλής προσφέροντας ανοχή στο επίπεδο της ποιότητας λειτουργίας της εφαρμογής.

Οι διεργασίες μιας εφαρμογής πραγματικού χρόνου μπορεί να προκαλούνται είτε ταυτόχρονα είτε προοδευτικά. Ο χρονοπρογραμματισμός ενός συνόλου διεργασιών απαιτεί την κατάρτιση ενός αυστηρού πλαισίου με σκοπό την τήρηση των χρονικών περιορισμών, που να λαμβάνει υπόψη:

την εκτέλεση όλων των διεργασιών όταν το σύστημα λειτουργεί με κανονικό τρόπο,

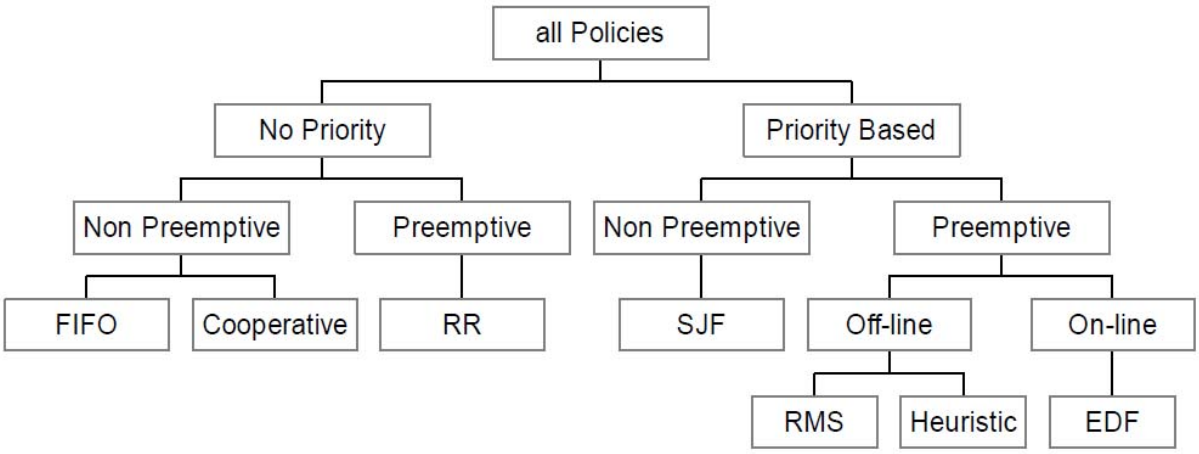
την εκτέλεση τουλάχιστον των διεργασιών που θεωρούνται ότι έχουν τον υψηλότερο βαθμό από άποψης σημαντικότητας όταν το σύστημα λειτουργεί με ανώμαλο τρόπο.

Οι αλγόριθμοι χρονοπρογραμματισμού αναθέτουν την εκτέλεση των διεργασιών στους επεξεργαστές και παρέχουν μια αυστηρά διατεταγμένη λίστα διεργασιών στην οποία

καθορίζεται η αλληλουχία και η προτεραιότητα εκτέλεσής τους. Οι αλγόριθμοι που χρησιμοποιούνται διακρίνονται σε:

Off-line: ένας αλγόριθμος off-line καταρτίζει μια ολοκληρωμένη λίστα με την αλληλουχία των διεργασιών και τις παραμέτρους που καθορίζουν την κάθε μία. Η σειρά εκτέλεσης των διεργασιών είναι εκ των προτέρων γνωστή. Η συγκεκριμένη προσέγγιση είναι στατική και άκαμπτη. Θεωρεί ότι όλες οι παράμετροι, όπως οι χρόνοι άφιξης των διεργασιών, οι χρόνοι εκτέλεσης, κλπ., είναι σταθεροί και έτσι δεν υπάρχει δυνατότητα προσαρμογής σε μεταβολές του εξωτερικού περιβάλλοντος.

On-line: η κατηγορία αυτή έχει γνώση των παραμέτρων όλων των διεργασιών που περιμένουν να εκτελεστούν και επιτρέπει κάθε χρονική στιγμή την επιλογή της επόμενης διεργασίας. Έτσι, ανάλογα με τα εξωτερικά γεγονότα, ο αλγόριθμός αποφασίζει για το ποια διεργασία θα ακολουθήσει αυτήν που εκτελείται. Αυτή η κατηγορία αλγορίθμων επιλέγεται όταν υπάρχει η ανάγκη για δυναμικό χειρισμό των διεργασιών και όταν υπάρχουν λίγες πληροφορίες για τη συμπεριφορά της υπό έλεγχο διαδικασίας. Χρησιμοποιείται κυρίως για χειρισμό σποραδικών διεργασιών και για συστήματα που θεωρείται σίγουρο ότι θα λειτουργήσουν με ανώμαλο τρόπο.



Σχήμα 3.2 Κατηγορίες αλγορίθμων χρονοπρογραμματισμού.

Προεκτοπίσιμοι (preemptive): Σε μια τέτοια κατηγορία αλγορίθμων χρονοπρογραμμματισμού, η εκτέλεση μιας διεργασίας μπορεί να διακοπεί βίαια πριν τον προβλεπόμενο χρόνο ολοκλήρωσής της και να ανατεθεί στον επεξεργαστή μια άλλη διεργασία με μεγαλύτερη προτεραιότητα. Η διεργασία που διακόπτεται μεταπίπτει στην κατάσταση ετοιμότητας και αναμένει για το πότε θα γίνει επιλεγμένη. Πλεονέκτημα αυτού του τύπου χρονοπρογραμματισμού είναι ότι με κατάλληλο χειρισμό των διεργασιών αποφεύγονται οι παραβιάσεις των χρονικών περιορισμών.

Μη προεκτοπίσιμοι (non preemptive): Κατά τον μη προεκτοπίσιμο χρονοπρογραμματισμό, οι διεργασίες από τη στιγμή που θα αρχίσουν να εκτελούνται, θα ολοκληρωθούν οπωσδήποτε. Μειονέκτημα της μεθόδου αυτής αποτελεί το γεγονός ότι το σύστημα ίσως να μην είναι σε θέση να τηρήσει τις χρονικές προθεσμίες.

Βέλτιστης προσπάθειας (best effort): Η τεχνική αυτή χρησιμοποιείται στα ήπια συστήματα όταν η εφαρμογή επιτρέπει ανοχή στην τήρηση των χρονικών περιορισμών.

Κατανεμημένοι: χρονοπρογραμματισμός τέτοιου είδους χρησιμοποιείται όταν οι παράμετροι όλων των διεργασιών είναι εκ των προτέρων γνωστοί και είναι δυνατό να δομηθούν σε μια κατανεμημένη αρχιτεκτονική. Η τεχνική αυτή χρησιμοποιείται όταν η εφαρ-

μογή αποτελείται από διάφορα επίπεδα και σε κάθε επίπεδο τηρείται μια «τοπική» λογική χρονοπρογραμματισμού.

3.1.5 Πολιτικές Εκτέλεσης Χρονοπρογραμματισμού.

Η εκτέλεση του χρονοπρογραμματισμού βασίζεται σε συμβατικές δομές δεδομένων:

Πίνακας επιλογής (election table): όταν ο χρονοπρογραμματισμός είναι σταθερή και γνωστή πριν την έναρξη της εφαρμογής, όπως για παράδειγμα στον στατικό off-line χρονοπρογραμματισμό, τότε μπορεί να αποθηκευτεί σε ένα πίνακα, που επιτρέπει στον χρονοπρογραμματιστή να γνωρίζει ποια είναι η επόμενη διεργασία προς εκτέλεση.

Λίστα προτεραιότητας : στον on-line χρονοπρογραμματισμό δημιουργείται μια δυναμική αλληλουχία διεργασιών, της οποίας η πρώτη στη σειρά διεργασία είναι αυτή που εκτελείται την κάθε χρονική στιγμή. Η δομή αυτή σύμφωνα με την οποία καταρτίζεται η λίστα της αλληλουχίας των διαδικασιών καλείται λίστα προτεραιότητας (priority queuing list ή priority ordered list).

Σταθερής ή μεταβλητής προτεραιότητας : η προτεραιότητα εκτέλεσης μιας διεργασίας καθορίζεται από ένα πλήθος χρονικών παραμέτρων. Η προτεραιότητα παραμένει σταθερή όταν οι αυτές οι παράμετροι, όπως για παράδειγμα ο χρόνος άφιξης και πέρατος, οι προθεσμίες, κλπ. δεν αλλάζουν. Η προτεραιότητα αυτή μεταβάλλεται όταν αυτοί οι παράγοντες αλλάζουν κατά τη διάρκεια της εκτέλεσης της διεργασίας.

Χρονοπρογραμματισμός δύο επιπέδων (two level scheduling): όταν ο χρονοπρογραμματισμός γίνεται πολύπλοκος, διαιρείται σε δύο μέρη. Το ένα επεξεργάζεται την πολιτική που θα ακολουθηθεί (υψηλού επιπέδου αποφάσεις, όπως την επιλογή της προτεραιότητας κάποιων κρίσιμων διεργασιών). Το άλλο μέρος επεξεργάζεται χαμηλότερου επιπέδου μηχανισμούς, όπως την εκτέλεση μιας διεργασίας που έχει αποφασιστεί να εκτελεστεί από το υψηλότερο επίπεδο

3.1.6 Μετρικές αξιολόγησης της απόδοσης.

Average response time: $$\overline{t_r} = \frac{1}{n}\sum_{i=1}^{n}(f_i - a_i)$$

Total completion time: $$t_c = \max_i(f_i) - \min_i(a_i)$$

Weighted sum of completion times: $$t_w = \sum_{i=1}^{n} w_i f_i$$

Maximum lateness: $$L_{max} = \max_i(f_i - d_i)$$

Maximum number of late tasks: $$N_{late} = \sum_{i=1}^{n} miss(f_i)$$

$$\text{where} \qquad miss(f_i) = \begin{cases} 0 & \text{if } f_i \le d_i \\ 1 & \text{otherwise} \end{cases}$$

Σχήμα 3.3 Μετρικές αξιολόγησης απόδοσης αλγορίθμων.

Η απόδοση των αλγορίθμων χρονοπρογραμματισμού αξιολογείται μέσω μια συνάρτηση κόστους που ορίζεται για το σύνολο των εργασιών. Για παράδειγμα, οι κλασικοί αλγόριθμοι προσπαθούν να ελαχιστοποιήσουν το μέσο χρόνο απόκρισης, το χρόνο ολοκλήρωσης, το σταθμισμένο άθροισμα των χρόνων ολοκλήρωσης, ή η μέγιστη καθυστέρηση. Όταν θεωρούνται σημαντικές οι προθεσμίες, συνήθως προσθέτουν περιορισμούς, επιβάλλοντας ότι όλες οι εργασίες πρέπει να τηρούν τις προθεσμίες τους. Αν κάποια προθεσμία δεν μπορεί να καλυφθεί με έναν αλγόριθμο Α, ο χρονοπρογραμματισμός λέγεται ότι είναι ανέφικτος με τον αλγόριθμο Α. Οι πιο σημαντικές συναρτήσεις κόστους[2] που χρησιμοποιούνται για την αξιολόγηση της απόδοσης ενός αλγορίθμου χρονοπρογραμματισμού φαίνονται στο σχήμα 3.3.

3.2 Χρονοπρογραμματισμός ανεξάρτητων περιοδικών διεργασιών.

Σε αυτή την παράγραφο περιγράφονται τέσσερεις βασικοί αλγόριθμοι χρονοπρογραμματισμού περιοδικών ανεξάρτητων διεργασιών, ο Rate Monotonic ή RM, ο inverse deadline ή DM, ο Earliest Deadline First ή EDF και ο Least Laxity First ή LLF, οι οποίοι χρησιμοποιούνται σε ένα ομογενές σύνολο περιοδικών διεργασιών.

Στον OnLine χρονοπρογραμματισμό ο αλγόριθμος χρονοπρογραμματισμού αναθέτει προτεραιότητες σύμφωνα με τις τρέχουσες παραμέτρους των διεργασιών. Εάν αυτές οι παράμετροι (χρόνοι άφιξης, προθεσμίες, κλπ.) είναι σταθερές και γνωστές από την αρχή, ο αλγόριθμος είναι στατικός, διαφορετικά είναι δυναμικός. Οι προτεραιότητες ανατίθενται στις διεργασίες πριν την έναρξη της λειτουργίας του συστήματος και δεν αλλάζουν κατά τη διάρκεια της εκτέλεσης. Οι βασικοί αλγόριθμοι για αυτού του είδους τον χρονοπρογραμματισμό είναι ο αλγόριθμος RM και ο αλγόριθμος DM. Εάν ο αλγόριθμος χρονοπρογραμματισμού πρέπει να βασιστεί σε μεταβλητές παραμέτρους, πρέπει να είναι δυναμικός, διότι η προτεραιότητα των διεργασιών μεταβάλλεται. Σε αυτή την κατηγορία οι σημαντικότεροι αλγόριθμοι είναι ο αλγόριθμος EDF και ο αλγόριθμος LLF.

3.2.1 Αλγόριθμος χρονοπρογραμματισμού Rate Monotonic

Ο αλγόριθμος χρονοπρογραμματισμού rate monotonic (RMS), εισήχθη για πρώτη φορά το 1973 από τους Liu και Leyland [7]. Ήταν ένας από τους πρώτους αλγορίθμους χρονοπρογραμματισμού για συστήματα πραγματικού χρόνου και χρησιμοποιείται ευρέως ακόμη και σήμερα. Ο RMS είναι αλγόριθμος στατικού χρονοπρογραμματισμού με σταθερές προτεραιότητες. Αποδεικνύεται ότι αυτές οι σταθερές προτεραιότητες αρκούν για να χρονοπρογραμματιστούν αποδοτικά οι διεργασίες σε πάρα πολλές περιπτώσεις.

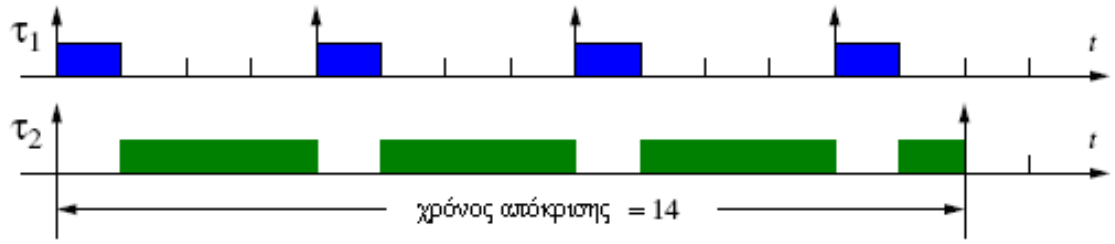
Η θεωρία του RMS ονομάζεται Rate Monotonic Analysis (RMA) και μελετά το θεωρητικό μοντέλο με τις παρακάτω απλές υποθέσεις:

- Όλες οι διεργασίες εκτελούνται περιοδικά σε μία και μόνη CPU.
- Ο χρόνος εναλλαγής περιβάλλοντος είναι μηδενικός και αγνοείται.
- Δεν υπάρχουν εξαρτήσεις δεδομένων μεταξύ των διεργασιών.
- Ο χρόνος εκτέλεσης κάθε διεργασίας είναι σταθερός.
- Η προθεσμία είναι στο τέλος της περιόδου.
- Εκτελείται η υψηλότερης προτεραιότητας έτοιμη διεργασία.

Το κύριο αποτέλεσμα της RMA είναι ότι μία σχετικά απλή πολιτική χρονοπρογραμματισμού είναι και βέλτιστη. Οι προτεραιότητες αποδίδονται με βάση την περίοδο. Η μεγαλύτερη προτεραιότητα δίνεται στη διεργασία με τη μικρότερη περίοδο. Για στατικές προ-

τεραιότητες είναι βέλτιστος αλγόριθμος εξασφαλίζοντας την υψηλότερη αξιοποίηση της CPU και ταυτόχρονα όλες οι διεργασίες ικανοποιούν τις προθεσμίες τους.

Στο Σχήμα 3.4 θεωρούμε ότι έχουμε δύο περιοδικές διεργασίες τ_1 και τ_2 με τις ακόλουθες παραμέτρους: $\tau_1(r_1, C, D, T) = (0, 1, 4, 4)$, $\tau_2(r_2, C, D, T) = (0, 10, 14, 14)$



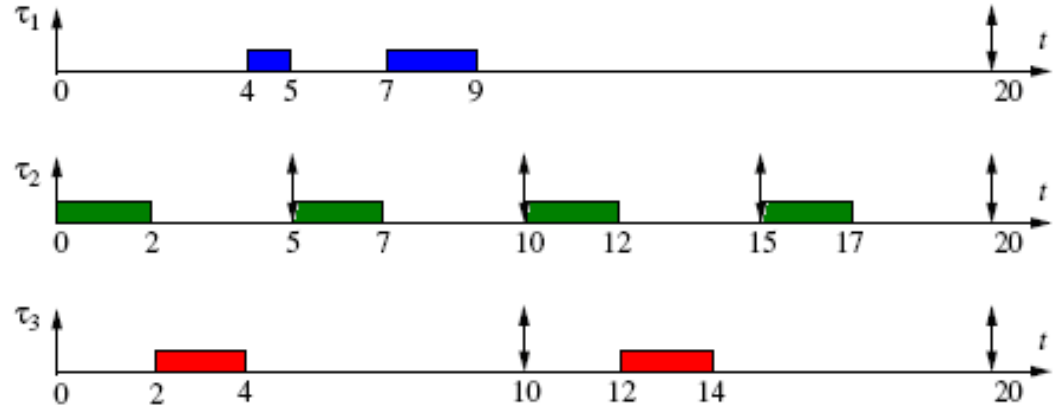
Σχήμα 3.4 Χρονοπρογραμματισμός κατά RMS των τ_1=(0,1,4,4) και τ_2=(0,10,14,14).

Οι Liu και Leyland απέδειξαν ότι ένα σύνολο που αποτελείται από n περιοδικές διεργασίες είναι δυνατό να χρονοπρογραμματιστεί εάν :

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n.(2^{\frac{1}{n}} - 1)$$

Για μεγάλες τιμές του n ($n \to \infty$) το πάνω όριο συγκλίνει στο ln(2). Μελέτες προσομοίωσης έδειξαν ότι για τυχαίο πλήθος διεργασιών, το όριο της χρήσης του επεξεργαστή ανέρχεται στο 88%.

Στο σχήμα 3.5 παρουσιάζεται ένα παράδειγμα χρονοπρογραμματισμού RM για ένα πλήθος από τρεις περιοδικές διεργασίες για κάθε μια από τις οποίες η σχετική τους προθεσμία είναι ίση με την περίοδό τους. Οι διεργασίες ορίζονται ως εξής: $\tau_1(0,3,20,20)$, $\tau_2(0,2,5,5)$ και $\tau_3(0,2,10,10)$. Η διεργασία τ_2 έχει την υψηλότερη προτεραιότητα και η διεργασία τ_1 την χαμηλότερη. Ο χρονοπρογραμματιστής εξετάζεται στον μέγιστο κύκλο μιας περιόδου, υπερπερίοδος του συνόλου των διεργασιών, που είναι το χρονικό διάστημα t = [0,20]. Ο συντελεστής χρήσης του επεξεργαστή είναι: $3/20+2/5+2/10 = 0.75 < 3(2^{1/3}-1) = 0.779$.



Σχήμα 3.5 Παράδειγμα χρονοπρογραμματισμού RM τριών διεργασιών τ_1=(0,3,20,20), τ_2=(0,2,5,5) και τ_3=(0,2,10,10)

Επειδή οι προτεραιότητες ανατίθενται ανάλογα με την περίοδο της κάθε διεργασίας, ο αλγόριθμος RM είναι χρήσιμο να πραγματοποιείται σε διεργασίες που οι σχετικές προθεσμίες τους είναι ίσες με τις περιόδους τους. Διαισθητικά αυτή η προτεραιοποίηση έχει νόημα καθώς τα έργα με τη μικρότερη περίοδο θα είναι αυτά που θα επανέλθουν πρώτα. Σε αυτήν την περίπτωση μπορεί να χρησιμοποιηθεί ο υπολογισμός του U για να διερευνηθεί ο χρονοπρογραμματισμού του συνόλου των εργασιών. Να σημειωθεί ότι η

συνθήκη είναι ικανή μόνο. Ο RM δεν είναι βέλτιστος όταν η προθεσμία κάθε έργου δεν είναι ίση με την περίοδο του. Σε διεργασίες που οι σχετικές τους προθεσμίες διαφέρουν από την περίοδό τους, χρησιμοποιείται περισσότερο ο αλγόριθμος αντίστροφης προθεσμίας.

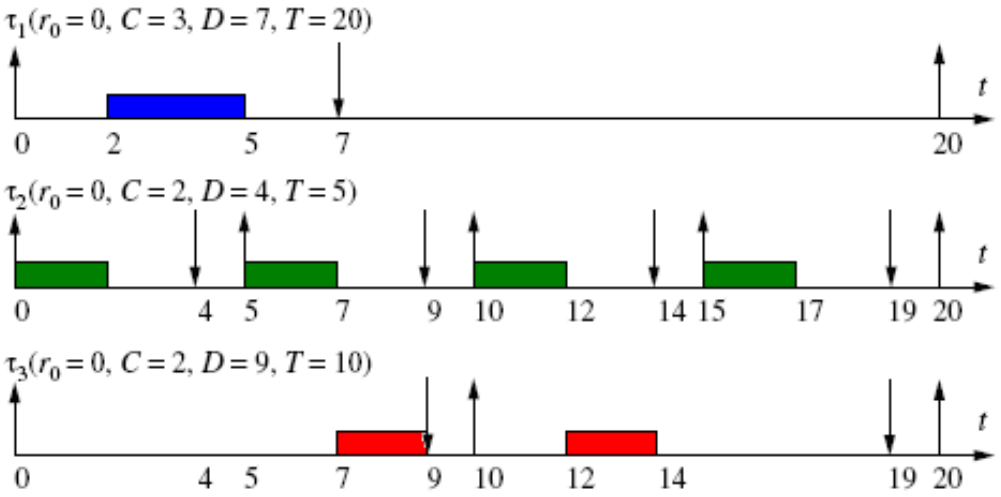
3.2.2 Αλγόριθμος αντίστροφης προθεσμίας (inverse deadline).

Ο αλγόριθμος αυτός δεν εστιάζει στη σχέση ισότητας μεταξύ της σχετικής προθεσμίας και της περιόδου των διεργασιών, αλλά αναθέτει προτεραιότητες λαμβάνοντας υπόψη μόνο τις σχετικές προθεσμίες των διεργασιών. Στις διεργασίες με τη μικρότερη προθεσμία ανατίθενται υψηλότερη προτεραιότητα. Διαισθητικά η ιδέα πίσω από τον DM είναι ότι το έργο με το μικρότερο χρονικό περιθώριο(όχι απαραίτητα αυτό με τη μικρότερη περίοδο) είναι το πιο επειγόμενο και έτσι του δίνει τη μεγαλύτερη προτεραιότητα. Ο αλγόριθμος αυτός είναι βέλτιστος.

Για κάθε αυθαίρετο πλήθος n διεργασιών με τις χρονικές προθεσμίες τους να είναι μικρότερες από τις περιόδους τους ισχύει η συνθήκη

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n.(2^{1/n} - 1)$$

Το Σχήμα 3.6 παρουσιάζει ένα παράδειγμα χρονοπρογραμματισμού αντίστροφης προθεσμίας για ένα πλήθος από τρεις περιοδικές διεργασίες με παραμέτρους τ_1(0,3,7,20), τ_2(0,2,4,5) και τ_3(0,2,9,10). Η διεργασία τ_2 έχει την υψηλότερη προτεραιότητα αφού έχει τη μικρότερη χρονική προθεσμία, ενώ η διεργασία τ_3 τη μικρότερη. Ο χρονοπρογραμματισμός είναι εφικτός και δίνεται σε ένα κύκλο της περιόδου του συνόλου των διεργασιών.



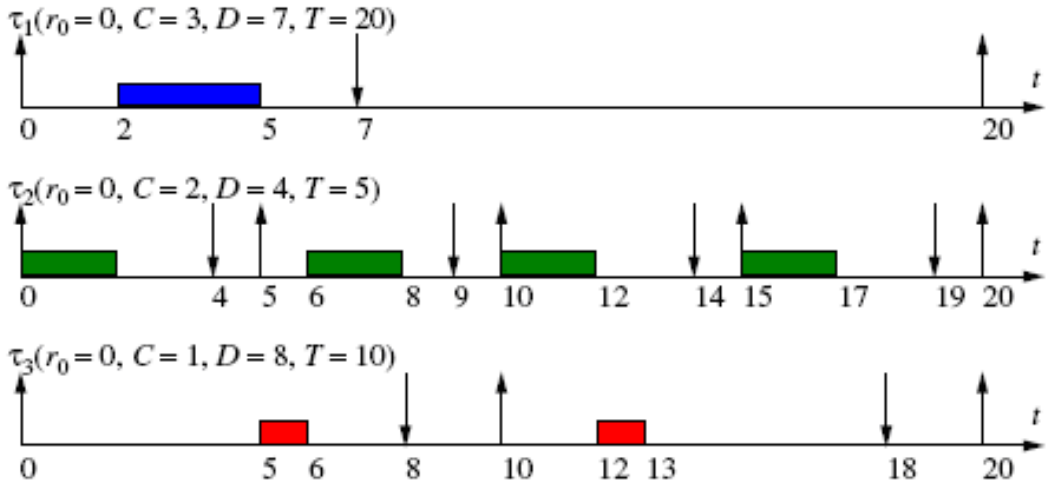
Σχήμα 3.6 Παράδειγμα χρονοπρογραμματισμού αντίστροφης προθεσμίας τριών διεργασιών τ_1=(0,3,7,20), τ_2=(0,2,4,5) και τ_3=(0,2,9,10)

3.2.3 Αλγόριθμος χρονοπρογραμματισμού Earliest Deadline First (EDF).

Ο αλγόριθμος χρονοπρογραμματισμού Earliest Deadline First (EDF) ανήκει στους δυναμικούς αλγόριθμους χρονοπρογραμματισμού χαρακτηριστικό των οποίων είναι η ανάθεση προτεραιοτήτων με δυναμικό τρόπο. Λαμβάνει υπόψη δηλαδή τις δυναμικές παραμέτρους των διεργασιών που αλλάζουν συνεχώς κατά τη διάρκεια της εκτέλεσή τους. Έτσι αυτός ο αλγόριθμος αναθέτει προτεραιότητες βασιζόμενος στις απόλυτες χρονικές προθεσμίες των διεργασιών. Η διεργασία που έχει κάθε στιγμή τη μικρότερη απόλυτη χρονική προθεσμία (absolute deadline) είναι αυτή με την υψηλότερη προτεραιότητα.

Η προτεραιότητα Pi ενός έργου την κάθε χρονική στιγμή t δίνεται από τον τύπο Pi=di(t)-t όπου di(t) είναι η επόμενη προθεσμία του ti (στο ή μετά το t). Ένα πλήθος από n περιοδικές διεργασίες με χρονικές προθεσμίες ίσες με τις περιόδους τους, χρονοπρογραμματίζονται με τον αλγόριθμο EDF αν και μόνο αν ο συντελεστής χρήσης του επεξεργαστή είναι μικρότερος ή ίσος της μονάδας:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$



Σχήμα 3.7 EDF χρονοπρογραμματισμός τριών διεργασιών τ_1=(0,3,7,20), τ_2=(0,2,4,5) και τ_3=(0,1,8,10).

Στο σχήμα 3.7 παρουσιάζεται ένα παράδειγμα χρονοπρογραμματισμού με τον EDF για τρεις περιοδικές διεργασίες με τ_1(0,3,7,20), τ_2(0,2,4,5) και τ_3(0,1,8,10). Τη χρονική στιγμή t=0, οι τρεις διεργασίες είναι έτοιμες προς εκτέλεση και η διεργασία που έχει αυτή τη χρονική στιγμή τη μικρότερη απόλυτη προθεσμία είναι η τ_2. Έτσι, η πρώτη που εκτελείται είναι η τ_2. Σε χρόνο t=2, η τ_2 ολοκληρώνει. Τώρα η διεργασία με τη μικρότερη απόλυτη χρονική προθεσμία είναι η τ1 η οποία και αρχίζει την εκτέλεσή της. Στη χρονική στιγμή t=5, η τ_1 ολοκληρώνει και η τ_2 είναι και πάλι σε ετοιμότητα. Όμως αυτή που θα εκτελεστεί είναι η τ3 διότι αυτή έχει τώρα τη μικρότερη χρονική προθεσμία.

3.2.4 Αλγόριθμος χρονοπρογραμματισμού Least Laxity First (LLF).

Είναι αλγόριθμος εξυπηρέτησης πρώτα της διεργασίας με τη μικρότερη χαλαρότητα (least laxity first). Αναθέτει προτεραιότητες στις διεργασίες ανάλογα με τη σχετική χαλαρότητά τους, δηλαδή το χρονικό περιθώριο που τους απομένει από τη χρονική στιγμή στην οποία βρίσκονται, μέχρι την ολοκλήρωση της εκτέλεσής τους εντός της χρονικής προθεσμίας τους Οι διεργασίες με τη μικρότερη χαλαρότητα (laxity), εκτελούνται με τη μεγαλύτερη προτεραιότητα. Το βέλτιστο του αλγορίθμου, ελέγχεται και ικανοποιείται σύμφωνα με τα όσα αναφέρθηκαν και στον αλγόριθμο EDF.

Στον LLF, όταν ένα έργο είναι έτοιμο, ο χρόνος έως την προθεσμία του μειώνεται καθώς ο χρόνος κυλάει. Ο γενικός τύπος που ορίζει τη χαλαρότητα ενός έργου i είναι:

Laxity(i,t)=max(T_D(i)-T_C(i)-t, 0.00) όπου

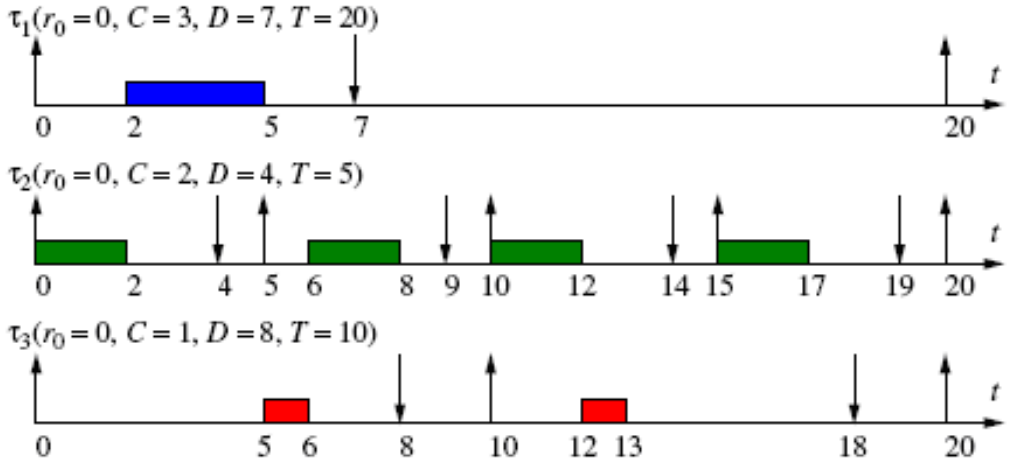
T_D(i) : η προθεσμία του έργου i

T_C(i) : ο χρόνος εκτέλεσης του έργου i

t : τρέχον χρόνος (πάντα θετικός).

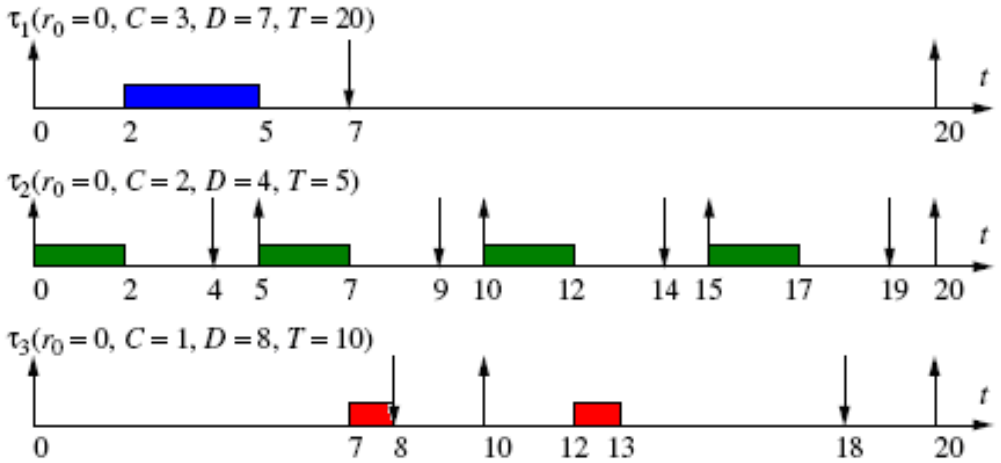
Είναι προφανές ότι η ελαστικότητα είναι δυναμικό χαρακτηριστικό και ο αλγόριθμος LLF είναι αλγόριθμος δυναμικού χρονοπρογραμματισμού.

Στο σχήμα 3.8 διακρίνεται ένα παράδειγμα χρονοπρογραμματισμού LLF ενός πλήθους τριών διεργασιών: $\tau_1(0,3,7,20)$, $\tau_2(0,2,4,5)$, και $\tau_3(0,1,8,10)$. Η σχετική χαλαρότητα των τριών διεργασιών υπολογίζεται κατά το χρόνο άφιξής τους. Τη χρονική στιγμή t=0, οι τρεις διεργασίες είναι έτοιμες προς εκτέλεση. Οι τιμές της σχετικής χαλαρότητάς τους είναι : $L(\tau_1) = 7 – 3 =4$, $L(\tau_2) = 4 – 2 = 2$, $L(\tau_3) = 8 – 1 = 7$. Η διεργασία με τη μικρότερη χαλαρότητα είναι η τ2 η οποία και εκτελείται πρώτη.



Σχήμα 3.8 Χρονοπρογραμματισμός LLF (τη χρονική στιγμή t=5, εκτελείται η διεργασία τ_3).

Τη χρονική στιγμή t=5, υπάρχει και πάλι αίτηση της τ_2 για εκτέλεσή της. Τη χρονική αυτή στιγμή όμως η σχετική χαλαρότητα της τ_2 είναι ίδια με αυτήν της τ_3: [$L(\tau_2) = 9\text{-}2 =7$ και $L(\tau_3) = 8\text{-}1 =7$. Επομένως διακρίνουμε δύο περιπτώσεις: είτε εκτελείται πρώτα η τ_3 σχήμα 3.8 είτε η τ_2 σχήμα 3.9.



Σχήμα 3.9 Χρονοπρογραμματισμός LLF (τη χρονική στιγμή t=5, εκτελείται η διεργασία τ_2).

3.3 Προσομοιωτές χρονοπρογραμματισμού.

Για να ελέγξουν τους χρονικούς περιορισμούς μίας εφαρμογής πραγματικού χρόνου που αποτελείται από πολλά ταυτόχρονα έργα οι σχεδιαστές συστημάτων πρέπει να ελέγξουν αν επιτυγχάνονται οι χρονικοί περιορισμοί των έργων. Από το 1980 πολλά μοντέλα, μέθοδοι και εργαλεία έχουν προταθεί για τον έλεγχο του αν ένα σύστημα πραγ-

ματικού χρόνου ικανοποιεί τις χρονικές του απαιτήσεις. Ένα από αυτά που λέγεται ανάλυση μονοτονικού ρυθμού είναι τμήμα ενός μεγαλύτερου συνόλου από ποσοτικές μεθόδους που λέγεται θεωρία χρονοπρογραμματισμού πραγματικού χρόνου. Αυτή η θεωρία βοηθά τους σχεδιαστές συστημάτων να προβλέψουν τη χρονική συμπεριφορά ενός συνόλου από έργα πραγματικού χρόνου με προσομοίωση χρονοπρογραμματισμού και δοκιμές εφικτότητας (feasibility) χρονοπρογραμματισμού. Η προσομοίωση χρονοπρογραμματισμού απαιτεί πρώτα τον υπολογισμό ενός χρονοπρογραμματισμού για ένα δοθέν χρονικό διάστημα και μετά να ψάξουμε για χρονικές ιδιότητες σ' αυτή. Αντίθετα οι δοκιμές εφικτού χρονοπρογραμματισμού επιτρέπουν στο σχεδιαστή να μελετήσει ένα σύνολο από εργασίες πραγματικού χρόνου χωρίς να γίνει η χρονοπρογραμματισμός.

Για την εγγύηση των χρονικών προθεσμιών των εργασιών έχουν προταθεί πολλές δοκιμές εφικτού χρονοπρογραμματισμού. Αυτές μπορούν να χωριστούν σε δοκιμές σε πολυωνυμικό χρόνο και σε λεπτομερείς δοκιμές. Οι δοκιμές σε πολυωνυμικό χρόνο μπορούν αν χρησιμοποιηθούν από τους δυναμικούς αλγορίθμους για να υπολογιστούν online οι απαιτούμενες εγγυήσεις και παρέχουν μόνο μία ικανή συνθήκη που μπορεί κάποιες φορές να οδηγήσει σε χαμηλή χρήση των πόρων. Οι λεπτομερείς δοκιμές παρέχουν μία συνθήκη αναγκαία και ικανή αλλά πολύ πολύπλοκη για online εκτέλεση ιδιαίτερα για μεγάλα σύνολα εργασιών αφού η πολυπλοκότητά τους είναι ψευδοπολυωνυμική.

Σαν συνέπεια της έντονής έρευνας σ' αυτό το χώρο προτάθηκαν πολλοί νέοι αλγόριθμοι. Παρά τον μεγάλο αριθμό τους όμως στα υπάρχοντα RTOS μόνο λίγοι αλγόριθμοι χρονοπρογραμματισμού πραγματικού χρόνου έχουν υλοποιηθεί. Μία μελέτη [8] έδειξε ότι σχεδόν όλα τα RTOS παρέχουν μόνο POSIX συμβατό χρονοπρογραμματισμό σταθερών προτεραιοτήτων. Έχει επίσης αποδειχτεί ότι ο χρονοπρογραμματισμός πραγματικού χρόνου με δυναμικές προθεσμίες οδηγεί σε αποτελεσματικότερη χρήση του επεξεργαστή και των άλλων πόρων του συστήματος.

Για να αξιολογηθεί ένας αλγόριθμος χρονοπρογραμματισμού πραγματικού χρόνου πρέπει να υλοποιηθεί σε ένα RTOS κάτι που είναι δύσκολο. Η προσομοίωση είναι ένας εναλλακτικός τρόπος για να αξιολογηθεί μία πολιτική χρονοπρογραμματισμού. Η απόδοση ενός αλγόριθμου χρονοπρογραμματισμού πραγματικού χρόνου μπορεί να εκτιμηθεί ευκολότερα με τη χρήση προσομοιωτή γι' αυτό το λόγο έχουν προταθεί αρκετοί προσομοιωτές.

Ο χρονοπρογραμματιστής είναι ένα τμήμα του πυρήνα του λειτουργικού συστήματος που υλοποιεί ένα σύνολο αλγορίθμων για να κατανείμει τους πόρους και τον έλεγχο πρόσβασης σε κοινόχρηστους πόρους μεταξύ των εργασιών [2]. Σε ένα RTOS οι υλοποιημένοι αλγόριθμοι πρέπει να είναι πραγματικού χρόνου.

Από την πλευρά της σχεδίασης τα RTS μπορούν να προσεγγιστούν από πολλές πλευρές. Για παράδειγμα οι μηχανικοί προτιμούν τον έλεγχο του υλικού ενώ οι επιστήμονες με το μοντέλο του συστήματος. Το μοντέλο του συστήματος θα εξηγήσει πως θα γίνει το μοντέλο αλληλεπίδρασης των εργασιών και πως θα αποδοθεί χρόνος επεξεργασίας σε κάθε εργασία. Η μοντελοποίηση του συστήματος είναι επαχθής επειδή υπάρχουν πολλές διαφορετικές πολιτικές χρονοπρογραμματισμού και το πρόβλημα του χρονοπρογραμματισμού είναι ένα ισχυρά πολύπλοκο πρόβλημα. Σαν συνέπεια της πολυπλοκότητας οι περισσότεροι που την μαθαίνουν αισθάνονται ότι η θεωρία χρονοπρογραμματισμού είναι μία σειρά από κανόνες που πρέπει να απομνημονευθούν. Έτσι δεν αποδίδεται η δέουσα προσοχή στο γεγονός ότι η πιο σημαντική έννοια δεν είναι η ακριβής περιγραφή ενός κανόνα αλλά ποιες συνθήκες και ποια προβλήματα είναι πιο κατάλληλα για κάθε κανόνα. Η ασάφεια στην κατανόηση μπορεί να λυθεί αποδίδοντας στις εργασίες όχι μόνο την λεπτομέρεια ενός χρονοπρογραμματισμού αλλά και τον πειραματισμό με

το πρόβλημα. Καίτοι αυτός μπορεί να γίνει με το χέρι έχει περιορισμό εξ' αιτίας της εκθετικής αύξηση στο χρόνο ανάλυσης με το μέγεθος του προβλήματος. Η χρήση της προσομοίωσης θα βοηθήσει να ξεπεραστεί αυτό το πρόβλημα.

Τέλος ένας άλλος σημαντικός στόχος της χρήσης των προσομοιωτών είναι σαν εκπαιδευτικά εργαλεία βοηθώντας τους εκπαιδευόμενους στα RTS να κατανοήσουν τις βασικές ιδέες που σχετίζονται με τη μοντελοποίηση των συστημάτων.

3.3.1 Προσομοιωτές χρονοπρογραμματισμού πραγματικού χρόνου.

Οι πρώτες συνεισφορές στη θεωρία του χρονοπρογραμματισμού έγιναν πριν 30 χρόνια από τους Liu και Leyland[7]. Η θεωρία επεκτάθηκε πολύ για να ανταποκριθεί στις απαιτήσεις των εφαρμογών και χρησιμοποιήθηκε με επιτυχία σε πολλά έργα.

Από την ακαδημαϊκή κοινότητα αναπτύχθηκαν κατά το παρελθόν αρκετά εργαλεία αλλά λίγα από αυτά φαίνεται ότι συνεχίζουν να συντηρούνται και να υποστηρίζονται μέχρι σήμερα. Τα περισσότερα από αυτά δεν παρέχουν και προσομοίωση και δοκιμή χρονοπρογραμματισμού.

Μία από τις πρώτες αξιόλογες προτάσεις ήταν ο προσομοιωτής που ονομαζόταν STRESS[9]. Ο N.C Audsley πρότειναν μία συλλογή από CASE εργαλεία για την ανάλυση και προσομοίωση της συμπεριφοράς αυστηρών συστημάτων πραγματικού χρόνου και εφαρμογών με υψηλές απαιτήσεις ασφάλειας. Ο προτεινόμενος προσομοιωτής περιλάμβανε μία γλώσσα προσομοίωσης για την περιγραφή των εργασιών του συστήματος και του περιβάλλοντος. Περιλάμβανε ένα γραφικό περιβάλλον επικοινωνίας για έλεγχο και απεικονίσεις, μερικές δοκιμές εφικτότητας και υποστήριξη για δίκτυα και συστήματα πολλών επεξεργαστών.

Μία άλλη πρόταση ήταν το FORTISSIMO[10] που βασιζόταν στο STRESS. Δεν ήταν μία εφαρμογή έτοιμη για χρήση αλλά ένα πλαίσιο εργασίας που επιτρέπει την ανάπτυξη προσομοιώσεων για χρονοπρογραμματισμό πραγματικού χρόνου. Παρείχε την βασική υποδομή για το χτίσιμο ενός προσομοιωτή κάποιου αλγορίθμου.

Το RTSIM[11] είναι ένα σύνολο εργαλείων προσομοίωσης χρονοπρογραμματισμού πραγματικού χρόνου με βασικό στόχο την εκπαίδευση. Είναι μία συλλογή από βιβλιοθήκες σε c/c++ για προσομοίωση συστημάτων πραγματικού χρόνου. Είναι ακόμη ενεργό πρόγραμμα και από το 2008 ο κώδικάς του έγινε διαθέσιμος.

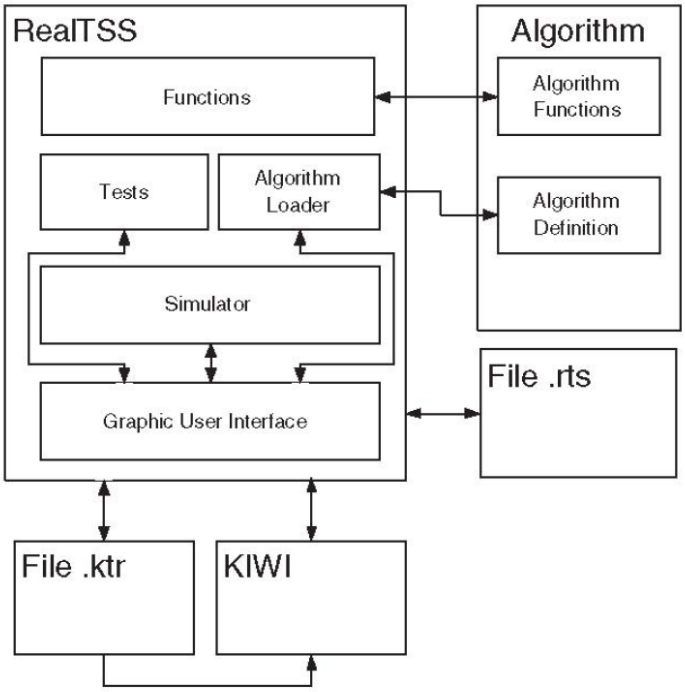
Στο [12] παρουσιάζεται μία εργασία για την χρήση μία αντικειμενοστρεφούς γλώσσα όπως η java στην ανάπτυξη προσομοιώσεων πραγματικού χρόνου. Οι συγγραφείς περίγραψαν τα πλεονεκτήματα αλλά και τα μειονεκτήματα της χρήσης της java για τον προσομοιωτή και έδωσαν μία επισκόπηση των απαραίτητων αλλαγών ώστε η Java να γίνει μία αποδεκτή επιλογή για τα συστήματα πραγματικού χρόνου.

Τέλος οι προσομοιωτές Rapid-RMA (http://www.tripac.com/rapid-rma) και TimeWiz (http://www.timesys.com) φαίνεται να είναι τα πιο πλήρη εργαλεία. Αυτά τα εργαλεία παρέχουν τους περισσότερους κλασικούς αλγορίθμους χρονοπρογραμματισμού και τις δοκιμές χρονοπρογραμματισμού. Μπορούν να συνδεθούν με διάφορα άλλα προγράμματα όπως εργαλεία CASE, μεσισμικό (middleware) και λειτουργικά συστήματα αλλά δυστυχώς δεν είναι διαθέσιμα δωρεάν.

3.3.2 Ο προσομοιωτής Realtss[13]

Το Realtss είναι ένας προσομοιωτής χρονοπρογραμματισμού πραγματικού χρόνου ανοικτού κώδικα, κατάλληλος να κάνει προσομοίωση ενός αλγορίθμου χωρίς να είναι αναγκαίο αυτός να υλοποιείται σε κάποιο RTOS. Το Realtss αναπτύχθηκε για να είναι ένα διδακτικό και ερευνητικό εργαλείο για υπάρχοντες και νέους αλγόριθμους χρονο-

προγραμματισμού πραγματικού χρόνου. Έχει σαν σκοπό να χρησιμοποιηθεί για την δοκιμή και την αξιολόγηση των αλγορίθμων χρονοπρογραμματισμού. Έχει γραφτεί σε TCL και διανέμεται σαν ανοικτό λογισμικό. Η αρθρωτή του σχεδίαση επιτρέπει την εύκολη προσθήκη νέων αλγορίθμων. Νέοι αλγόριθμοι χρονοπρογραμματισμού μπορούν να προστεθούν σαν τμήματα γραμμένα σε TCL, C, C++ και δεν περιορίζεται από καμία γλώσσα προσομοίωσης. Μπορεί να εκτελεστεί σε οποιοδήποτε λειτουργικό σύστημα (Linux, Windows).



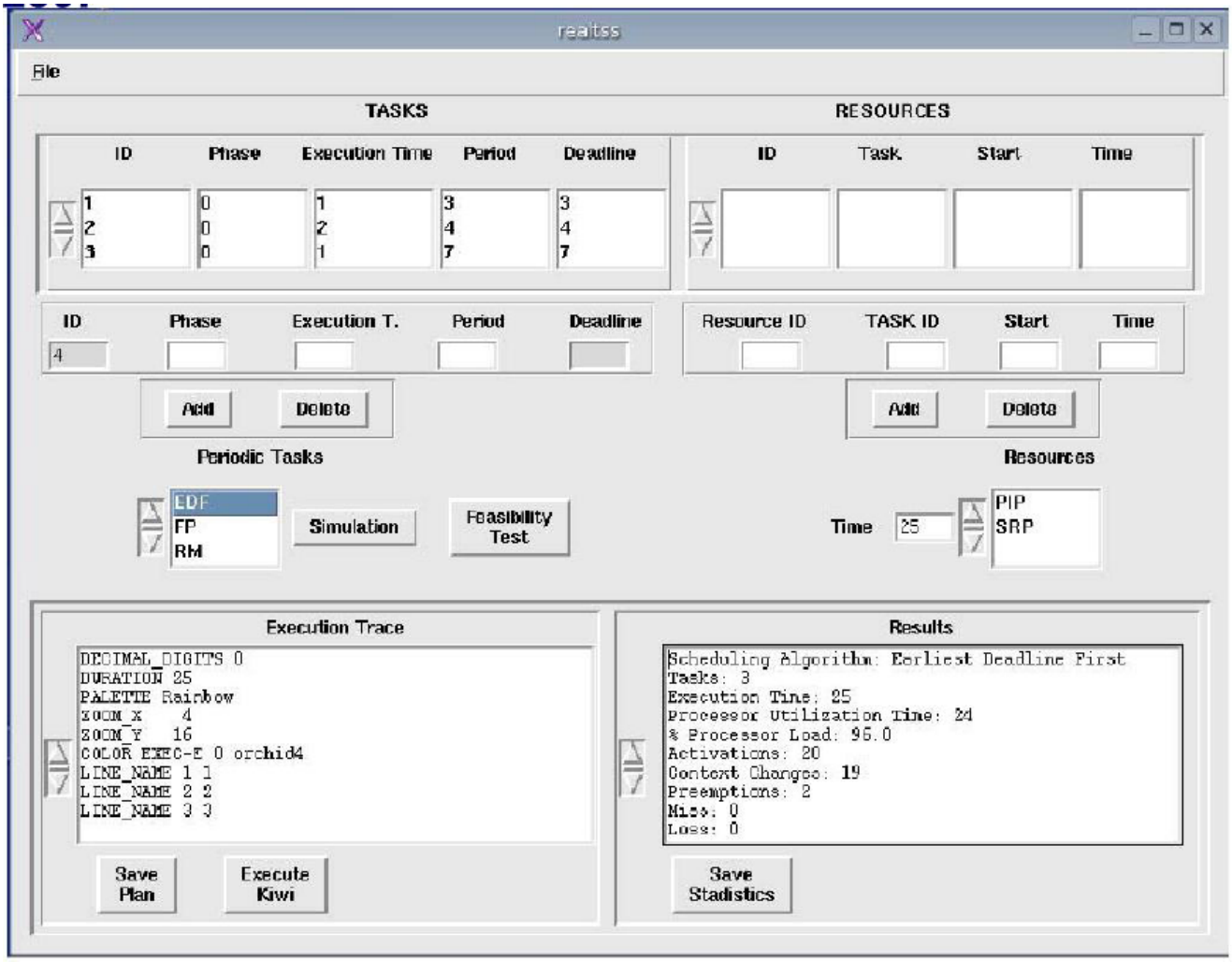
Σχήμα 3.10 Αρχιτεκτονική του Realtss.

Η αρχιτεκτονική του δομή φαίνεται στο σχήμα 3.10. Έχει αρθρωτή δομή. Το τμήμα function υλοποιεί ένα σύνολο λειτουργιών για το οριζόμενο σύνολο εργασιών. Το τμήμα algorithm loader αναζητεί και φορτώνει τον κατάλληλο χρονοπρογραμματιστή από το πλαίσιο εργασίας. Για να προσθέσει ο χρήστης ένα νέο αλγόριθμο ακολουθεί τις οδηγίες του οδηγού χρήστη και αποθηκεύει το τμήμα του στον φάκελο jalgos. Ο προσομοιωτής αλγορίθμου αποτελείται από δύο αρχεία. Το πρώτο είναι ένα αρχείο tcl που περιέχει τον προσομοιωτή και μπορεί να χρησιμοποιήσει οποιαδήποτε λειτουργία είναι ήδη διαθέσιμη στο Realtss. Το δεύτερο είναι ένα αρχείο det που περιέχει τους ορισμούς των αλγορίθμων που θα χρησιμοποιηθούν από το Realtss. Το τμήμα tests περιέχει δοκιμαστικά δεδομένα για του αλγορίθμους που υλοποιούνται. Νέα σύνολα εργασιών για δοκιμή μπορούν να προστεθούν εύκολα. Οι παράμετροι των δοκιμαστικών δεδομένων μπορούν να αποθηκευτούν στο rts αρχείο για να χρησιμοποιηθούν αργότερα.

Μετά την εκτέλεση της προσομοίωσης δημιουργούνται μερικά νέα αρχεία. Ένα από αυτά περιλαμβάνει τα αποτελέσματα της προσομοίωσης που θα εμφανιστούν στη γραφική διεπαφή του Realtss. Το άλλο είναι ένα αρχείο ktr που με τη σειρά του χρησιμοποιείται από το KIWI. Το Realtss είναι πλήρως ολοκληρωμένο με το KIWI[14] που είναι μία γραφική εφαρμογή που παρουσιάζει τα logs της εκτέλεσης του έργου. Τα αποτελέσματα της προσομοίωσης παράγονται σε ένα KIWI συμβατό format για να εμφανιστούν κατάλληλα με τη χρήση του. Μετά την εκτέλεση της δοκιμής της προσομοίωσης ο χρήστης μπορεί να αναλύσει τη συμπεριφορά του συνόλου του έργου. Η χρονική συμπεριφορά της προσομοίωσης μπορεί να οπτικοποιηθεί με το KIWI.

Η χρήστη του προσομοιωτή έχει τρία βήματα: ορισμός παραμέτρων, εκτέλεση της προσομοίωσης και ανάλυση των αποτελεσμάτων. Στο πρώτο βήμα ο χρήστης ορίζει τις παραμέτρους των εργασιών. Οι παράμετροι που μπορούν να οριστούν είναι η περίοδος, ο χρόνος εκτέλεσης (στη χειρότερη περίπτωση), η φάση και η προθεσμία.

Το σύστημα πραγματικού χρόνου μπορεί να αποτελείται από εργασίες με αυστηρές, από εργασίες με ήπιες προθεσμίες και από μοιραζόμενους πόρους. Έτσι αν το σύστημα απαιτεί τη χρήση πόρων όπως mutexs οι παράμετροι των πόρων μπορούν επίσης να οριστούν κατάλληλα. Στο σχήμα 3.11 φαίνεται η γραφική διεπαφή του Realtss.



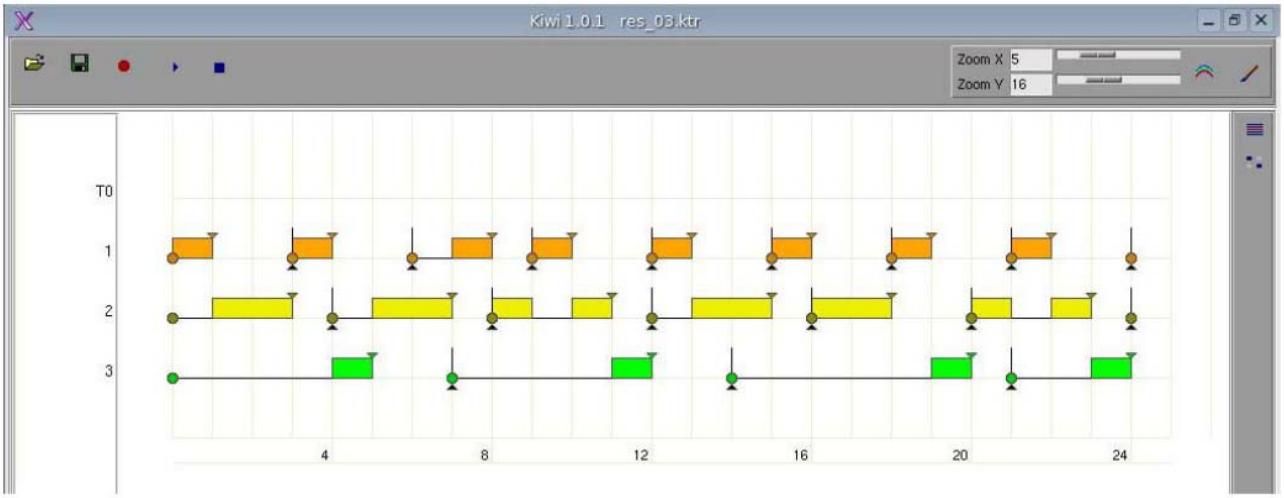
Σχήμα 3.11 Διεπαφή χρήστη στο Realtss.

Ήδη το Realtss περιλαμβάνει τους αλγόριθμους χρονοπρογραμματισμού:

- POSIX-complaint Fixed-Priorities.
- Rate Monotonic (RM).
- Earliest Deadline First (EDF).
- Deadline Monotonic (DM).
- Priority Inheritance Protocol (PIP).
- POSIX-complaint version of the Priority Ceiling Protocol(PCP).
- Stack Resource Protocol (SRP).

Οι πολιτικές εμφανίζονται σε μία λίστα από την οποία ο χρήστης επιλέγει την επιθυμητή. Ένας απλός μηχανισμός επιτρέπει την προσθήκη νέων πολιτικών χρονοπρογραμματισμού. Νέα τμήματα μπορούν να προστεθούν και να εμφανίζονται στη λίστα επιλογής του προσομοιωτή.

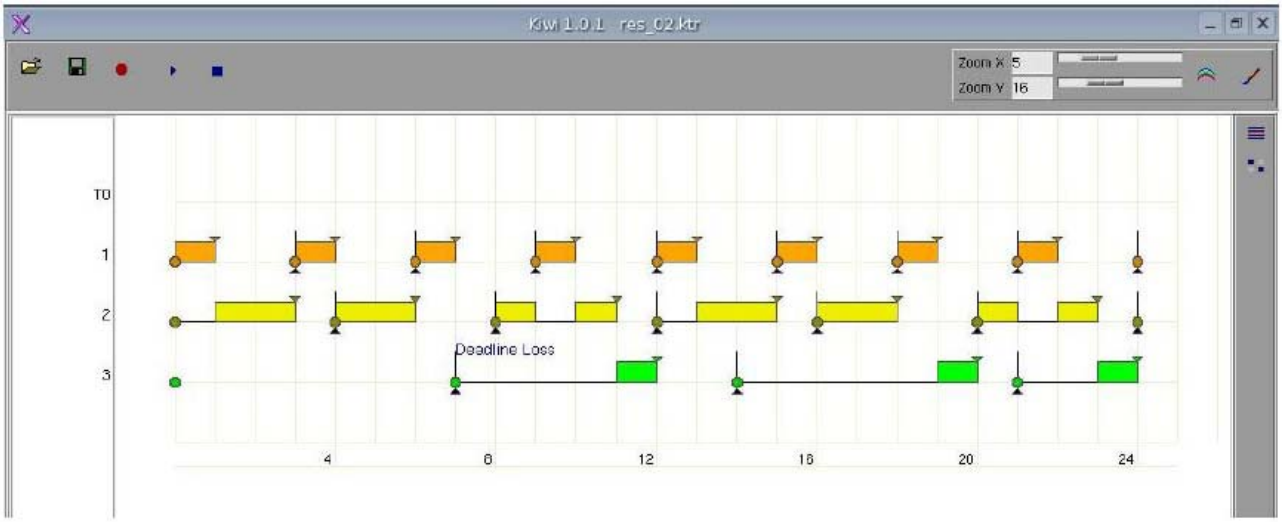
Αφού οριστούν οι παράμετροι και επιλεγεί η επιθυμητή πολιτική χρονοπρογραμματισμού μπορούν να εκτελεστούν μία δοκιμή χρονοπρογραμματισμού και μία προσομοίωση για κάποιο δοθέν χρονικό διάστημα.



Σχήμα 3.12 Χρονοπρογραμματισμός EDF.

Μαζί με τα δεδομένα που απεικονίζονται γραφικά επιπλέον χρήσιμες πληροφορίες παρέχονται από το Realtss. Αυτές είναι χρήσιμες για την αξιολόγηση της απόδοσης του χρησιμοποιηθέντος αλγορίθμου. Η παρεχόμενη πληροφόρηση δίνει τη χρήση του επεξεργαστή, το χρόνο απόκριση, το χρόνο αναμονής, τις χαμένες προθεσμίες και τα overruns. Οι παράμετροι της προσομοίωσης και τα αποτελέσματα μπορούν να αποθηκευτούν για να γίνει συγκριτική αξιολόγηση διαφορετικών αλγορίθμων χρονοπρογραμματισμού.

<u>Παράδειγμα.</u>



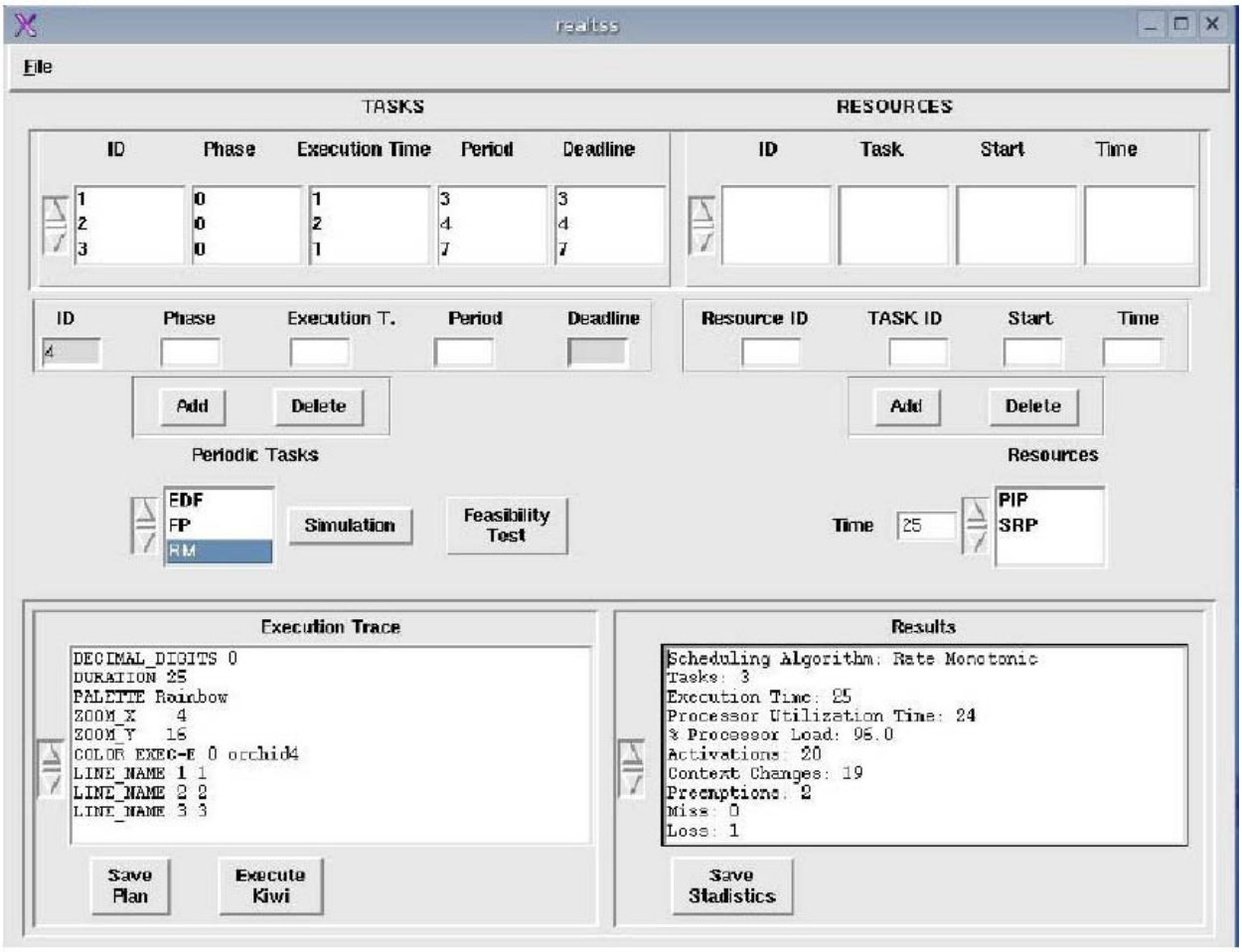
Σχήμα 3.13 Χρονοπρογραμματισμός RM.

Στο σχήμα 3.11 φαίνεται ένα παράδειγμα. Είναι ένα σύστημα που αποτελείται από τρία έργα. Οι παράμετροι κάθε έργου είναι τ1=(3,1), τ2=(5,2) και τ3=(8,1). Ο πρώτος αριθμός αντιστοιχεί στην περίοδο και ο δεύτερος στο χρόνο εκτέλεσης. Η πολιτική χρονοπρο-

γραμματισμού για το παράδειγμα είναι η EDF. Ο χρόνος προσομοίωσης στο παράδειγμα είναι 25 χρονικές μονάδες. Τα αποτελέσματα της προσομοίωσης απεικονίζονται στα σχήματα 3.12 και 3.13.

Το Realtss μπορεί να χρησιμοποιηθεί για την αξιολόγηση αλγορίθμων χρονοπρογραμματισμού και να συγκρίνει διαφορετικές πολιτικές. Το σύνολο των έργων του προηγούμενου παραδείγματος χρονοπρογραμματίστηκε και με τον αλγόριθμο RM. Το σχήμα 3.14 δείχνει τα αποτελέσματα της προσομοίωσης. Φαίνεται ότι ένα έργο χάνει μία φορά την προθεσμία του. Αυτό φαίνεται στο σχήμα 3.13 όπου το τ3 δεν εκτελέστηκε στην πρώτη ενεργοποίηση του και συνεπώς έχασε την προθεσμία του. Πιστοποιεί κανείς πειραματικά αυτό που γνωρίζει από τη θεωρία ότι ο EDF μπορεί να χρονοπρογραμματίσει έργα που δεν μπορεί ο RM.

Όπως φάνηκε από το παράδειγμα το realtss μπορεί να χρησιμοποιηθεί σαν ερευνητικό εργαλείο που επιτρέπει την αξιολόγηση της απόδοσης ενός αλγορίθμου χρονοπρογραμματισμού και τη σύγκριση αλγορίθμων μεταξύ τους χρησιμοποιώντας ίδια ή διαφορετικά δεδομένα. Επιπλέον μπορεί να χρησιμοποιηθεί και σαν εκπαιδευτικό εργαλείο. Για παράδειγμα μπορεί να δει κάποιος σε γραφική μορφή τις διαφορές ενός στατικού από ένα δυναμικό χρονοπρογραμματισμό και να τις συγκρίνει. Στο παράδειγμα μπορεί να δει κανείς ότι στον στατικό χρονοπρογραμματισμό οι προτεραιότητες δεν αλλάζουν. Από την άλλη όταν χρησιμοποιείται ένας δυναμικός αλγόριθμος οι προτεραιότητες μπορεί να αλλάξουν. Στο σχήμα 3.13 η τ1 πάντα προεκτοπίζει τις τ2 και τ3. Η τ1 έχει πάντα τη μέγιστη προτεραιότητα. Αντίθετα στο σχήμα 3.12 φαίνεται ότι το τ2 εκτοπίζει το τ1 στην τρίτη εκτέλεση επειδή έχει μεγαλύτερη προτεραιότητα σε εκείνη τη χρονική στιγμή. Η τ1 δεν έχει πάντα τη μέγιστη προτεραιότητα.

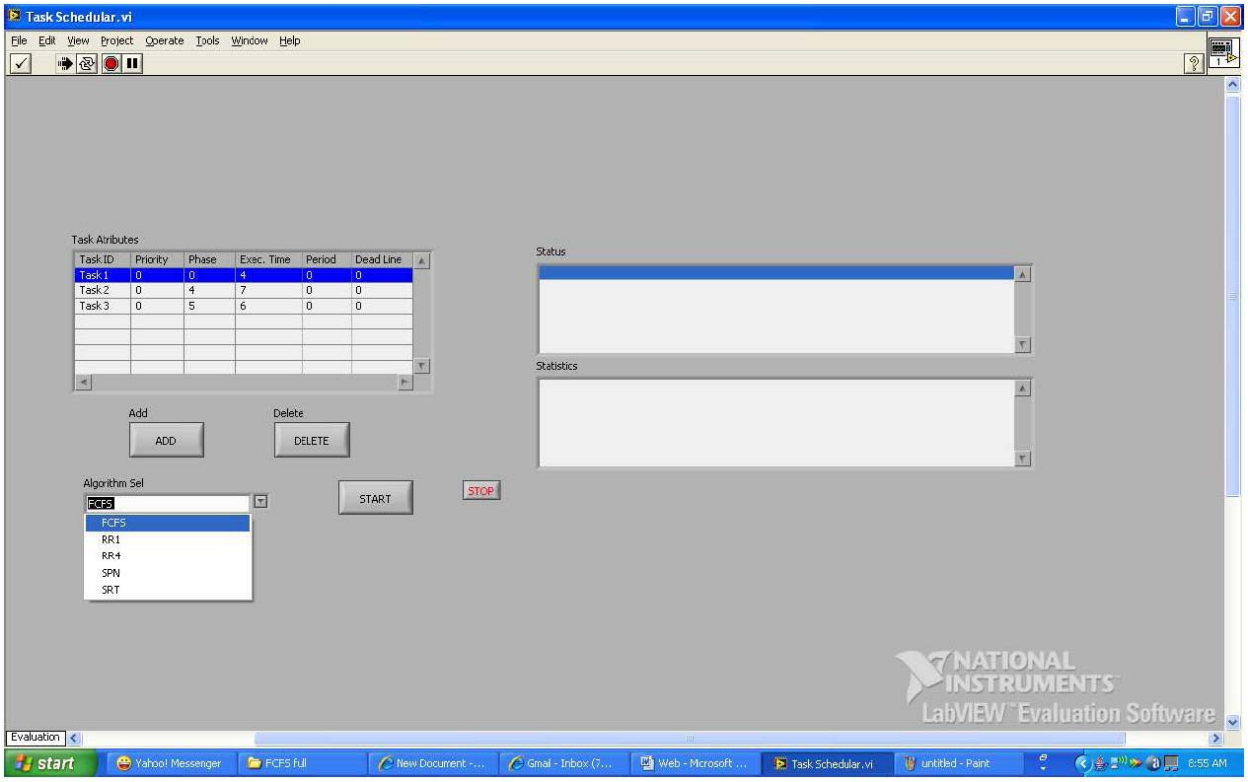


Σχήμα 3.14 Χρονοπρογραμματιστής RM**.**

3.3.3 AU Real-time Scheduler Simulator[15].

Το AUTRSS (AU Real-time Scheduler Simulator) σχεδιάστηκε για να χρησιμοποιηθεί σαν εκπαιδευτικό εργαλείο για τον χρονοπρογραμματισμό πραγματικού χρόνου και την αξιολόγηση πολιτικών χρονοπρογραμματισμού πραγματικού χρόνου που χρησιμοποιούνται στις ενσωματωμένες εφαρμογές πραγματικού χρόνου. Αναπτύχθηκε σαν πλαίσιο εργασίας στο LabVIEW.

Ένα πλαίσιο εργασίας για αξιολόγηση αλγορίθμων χρονοπρογραμματισμού πρέπει να ικανοποιεί χαρακτηριστικά όπως απλότητα, συμβατότητα με τα PC, συμβατότητα με το χρησιμοποιούμενο λειτουργικό σύστημα, ακρίβεια αποτελεσμάτων και ευκολία χρήσης. Η πλειονότητα αυτών των χαρακτηριστικών στοχεύει για χρήση στο εικονικό προσαρμοστικό χρήστη που φαίνεται στο σχήμα 3.15. Ο προτεινόμενος προσομοιωτής που βασίζεται στο web μπορεί να χρησιμοποιηθεί από ένα πρόγραμμα πλοήγησης και ένα σύνολο από κλικ στο παράθυρο εισαγωγής δεδομένων.



Σχήμα 3.15 Προσομοιωτής AUTRSS.

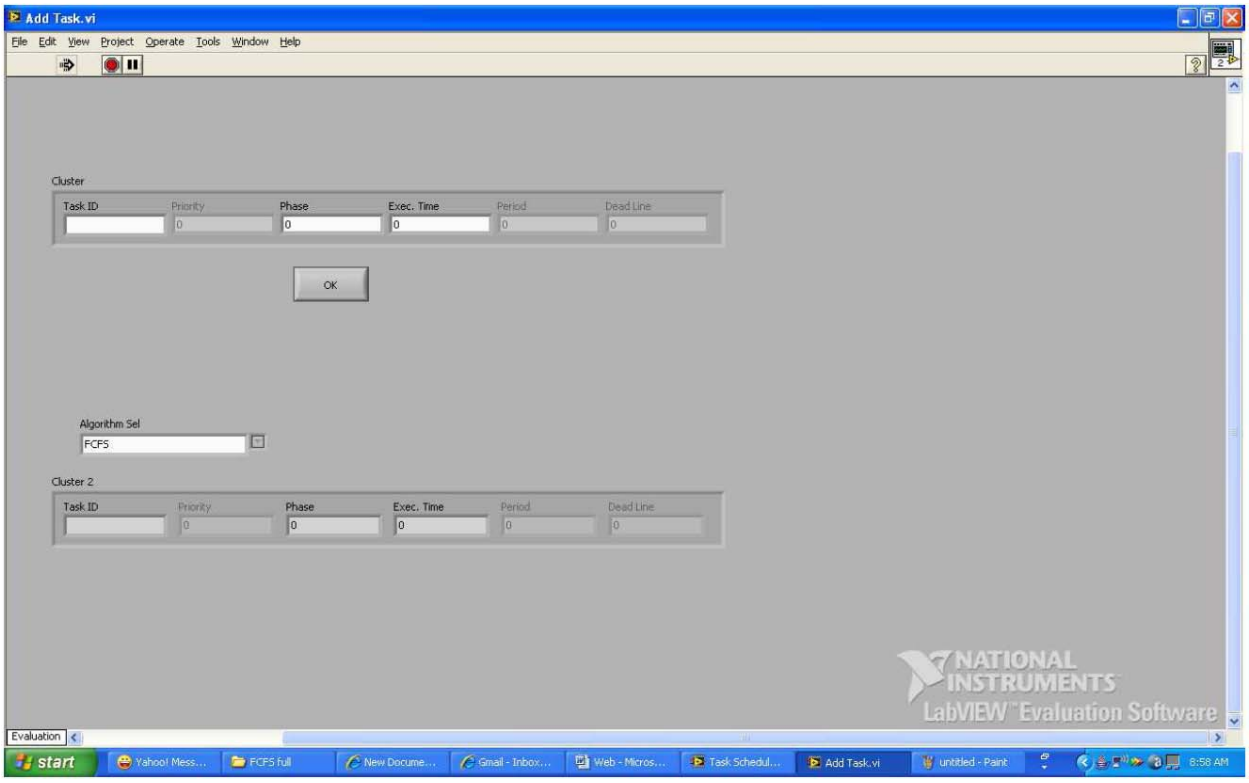
Η προσομοίωση. Το εργαλείο αξιολόγησης και ανάλυσης του αλγορίθμου εκτελεί τις ακόλουθες εργασίες: ορισμός έργου, δημιουργία συνόλου έργων, εκτέλεση επιλεγέντος αλγορίθμου, εκτέλεση της ανάλυσης αποτελεσμάτων και εμφάνιση των αποτελεσμάτων. Η αξιολόγηση της απόδοσης των αλγορίθμων βγαίνει βάση των αποτελεσμάτων που αποκτούνται από τους υπολογισμούς της ανάλυσης. Τα διάφορα στάδια της διαδικασίας αξιολόγησης είναι:

- Αναγνώριση των έργων
- Επιλογή του αλγορίθμου
- Χρονικό διάγραμμα προσομοίωσης
- Εκτέλεση προσομοίωσης.

Ο πιο αποδοτικός αλγόριθμός χρονοπρογραμματισμού, για δρομολόγηση έργων πραγματικού χρόνου, είναι αυτός που οδηγεί στον ελάχιστο χρόνο απόκρισης, τον ελάχιστο

αριθμό έργων που χάνουν τις προθεσμίες τους και οδηγούν στη μέγιστη χρήση των πόρων του συστήματος και με άλλες παραμέτρους.

Το πλήρες μοντέλο εργασίας είναι πολύ πολύπλοκο για υλοποίηση και μερικές από τις παραμέτρους αγνοούνται. Στα RTS δύο χαρακτηριστικά του έργου θεωρούνται πρωτευούσης σημασίας: η κρισιμότητα του έργου και ο χρόνος. Η κρισιμότητα της εργασίας είναι συχνά και υποκειμενική αντίθετα ο χρόνος είναι αντικειμενικό κριτήριο. Τα χρονικά χαρακτηριστικά ενός έργου είναι η προθεσμία του(TD), η περίοδός του(TP) και ο χρόνος εκτέλεσης στη χειρότερη περίπτωση(TCw).



Σχήμα 3.16 Προσομοιωτής AUTRSS.

Τα στοιχεία της προσομοίωσης. Τα χαρακτηριστικά του έργου είναι τμήμα του προσομοιωτή που επιτρέπει στο χρήστη να προσθέτει εργασίες – στο υπάρχον σύνολο – με παραμέτρους:

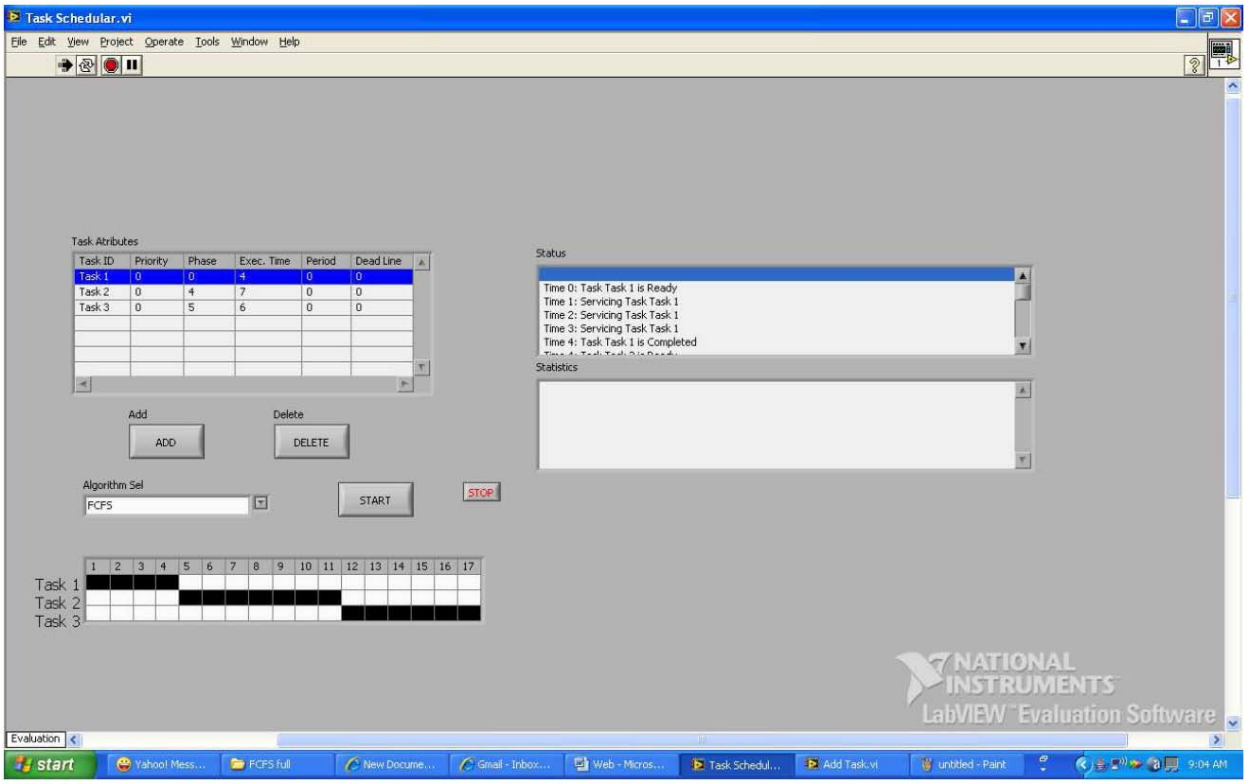
- όνομα έργου
- προτεραιότητά του
- φάση
- χρόνος εκτέλεσης
- περίοδος
- προθεσμία

Το όνομα έργου πρέπει να είναι μοναδικό για κάθε έργο. Όλες οι άλλες παράμετροι είναι αριθμητικές. Όλες οι παράμετροι απαιτούνται για να αποθηκεύσετε ένα έργο σε ένα σύνολο έργων, να τροποποιήσετε τις παραμέτρους ενός έργου, να διαγράψετε ένα έργο. Αυτά απεικονίζονται στο σχήμα 3.15.

Χρήση πόρων. Προσθέτει ένα πόρο για ένα συγκεκριμένο έργο ή έργα για το κρίσιμο τμήμα με όνομα πόρου, όνομα έργου, χρόνος έναρξης, χρόνος εκτέλεσης. Το σχήμα 3.16 απεικονίζει την προσθήκη – διαγραφή έργων.

Έλεγχος προσομοιωτή. Το τμήμα ελέγχου του προσομοιωτή είναι το βασικό τμήμα που παρέχει τις ακόλουθες λειτουργίες στον χρήστη:

- Επιλογή πολιτικής χρονοπρογραμματισμού
- Επιλογή πρωτοκόλλου συγχρονισμού των έργων
- Επιλογή πρεοκτοπισμού ή όχι
- Εκτέλεση δοκιμής χρονοπρογραμματισμού για να βρεθεί αν το σύνολο είναι χρονοπρογραμματίσιμο.



Σχήμα 3.17 Προσομοιωτής AUTRSS.

Ένας χρονοπρογραμματιστής πραγματικού χρόνου δίνει το χρόνο επεξεργασίας στα έργα σε διακριτά κβάντα. Κάθε έργο έχει μια βασική στατική προτεραιότητα παρότι κατά την εκτέλεση μπορεί κάποια στιγμή να αλλάξει την προτεραιότητά του έναντι των υπόλοιπων έργων. Στο χρόνο εκτέλεσης κάθε έργο ενεργοποιείται όσες φορές χρειάζεται. Ο χρόνος στον οποίο ένα έργο αρχίζει την ενεργοποίησή του και γίνεται έτοιμο για εκτέλεση είναι ο χρόνος άφιξης. Ο χρόνος που η ενεργοποίηση του έργου πραγματικά αρχίζει την εκτέλεσή του λέγεται χρόνος έναρξης. Αυτός ο χρόνος μπορεί να είναι διαφορετικός από το χρόνο άφιξης εάν για παράδειγμα ένα έργο υψηλότερης προτεραιότητας ήταν ήδη σε εκτέλεση. Ο χρόνος που το έργο ολοκληρώνει την εκτέλεση του είναι ο χρόνος τελειώματος. Οι ενεργοποιήσεις ενός έργου I συνήθως υποτίθεται ότι φτάνουν κανονικά με περίοδο Ti (Σποραδικά ή απεριοδικά έργα φτάνουν σε ακανόνιστα χρονικά διαστήματα αλλά με μία γνωστή ελάχιστη απόσταση). Για κάθε έργο ο προγραμματιστής ορίζει μια προθεσμία Di στην οποία κάθε ενεργοποίηση οφείλει να έχει ολοκληρωθεί. Για να υποστηρίξει την ανάλυση ο προγραμματιστής ορίζει και το χρόνο εκτέλεσης στη χειρότερη περίπτωση Ci. Υποθέτουμε ότι ο χρόνος εκτέλεσης περιέχει και την επιβάρυνση λόγω εναλλαγή περιβάλλοντος. Έτσι στο μοντέλο φαίνεται η εναλλαγή να είναι ακαριαία. Η διαφορά μεταξύ του της ολοκλήρωσης ενός έργου και της άφιξής του λέγεται χρόνος απόκρισης Ri. Κάθε εργασία κανονικά αρχίζει να εκτελεί την ενεργοποίησή της από το χρόνο 0 αλλά μπορεί και να καθυστερήσει οπότε έχουμε και ένα offset Oi.

Σημαντικότητα της Web υλοποίησης. Οι βελτιώσεις είναι άμεσα διαθέσιμες. Όποια στιγμή, οπουδήποτε, ένας χρήστης με πρόσβαση στο Διαδίκτυο μπορεί να χρησιμοποιήσει το εργαλείο για εκπαίδευση. Επειδή δεν απαιτεί επί πλέον υλικό ή λογισμικό είναι εύκολη η χρήση του.

Το σχήμα 3.17 δείχνει την έξοδο του προσομοιωτή. Το όνομα έργου, η προθεσμία, η προτεραιότητα, η περίοδος, ο χρόνος εκτέλεσης και η φάση είναι οι ιδιότητες του έργου που συμβάλουν στην προσομοίωση. Το χρονικό διάγραμμα της σε πραγματικό χρόνο εκτέλεση των εργασιών και τα στατιστικά στοιχεία του χρονοπρογραμματισμού είναι η παραγόμενη έξοδος. Το προτεινόμενο πλαίσιο εργασίας για προσομοίωση αλγορίθμων χρονοπρογραμματισμού κυρίως αναπτύχθηκε για να χρησιμοποιηθεί ως εργαλείο διδασκαλίας και αξιολόγηση της απόδοσης χρονοπρογραμματιστών πραγματικού χρόνου. Τα κριτήρια αξιολόγησης είναι με βάση το μέσο χρόνο απόκρισης, τον αριθμό των έργων που χάνουν την προθεσμία τους, τη χρήση του επεξεργαστή, τον αριθμό των προεκτοπίσεων και τις εναλλαγές πλαισίου. Η Web-based εγκατάσταση του προσομοιωτή δίνει τη δυνατότητα στο χρήστη να τη χρησιμοποιήσει χωρίς να εξαρτάται από μια πλατφόρμα, κάποιο υλικό ή κάποιο λογισμικό.

3.3.4 Ο προσομοιωτής Cheddar[16].

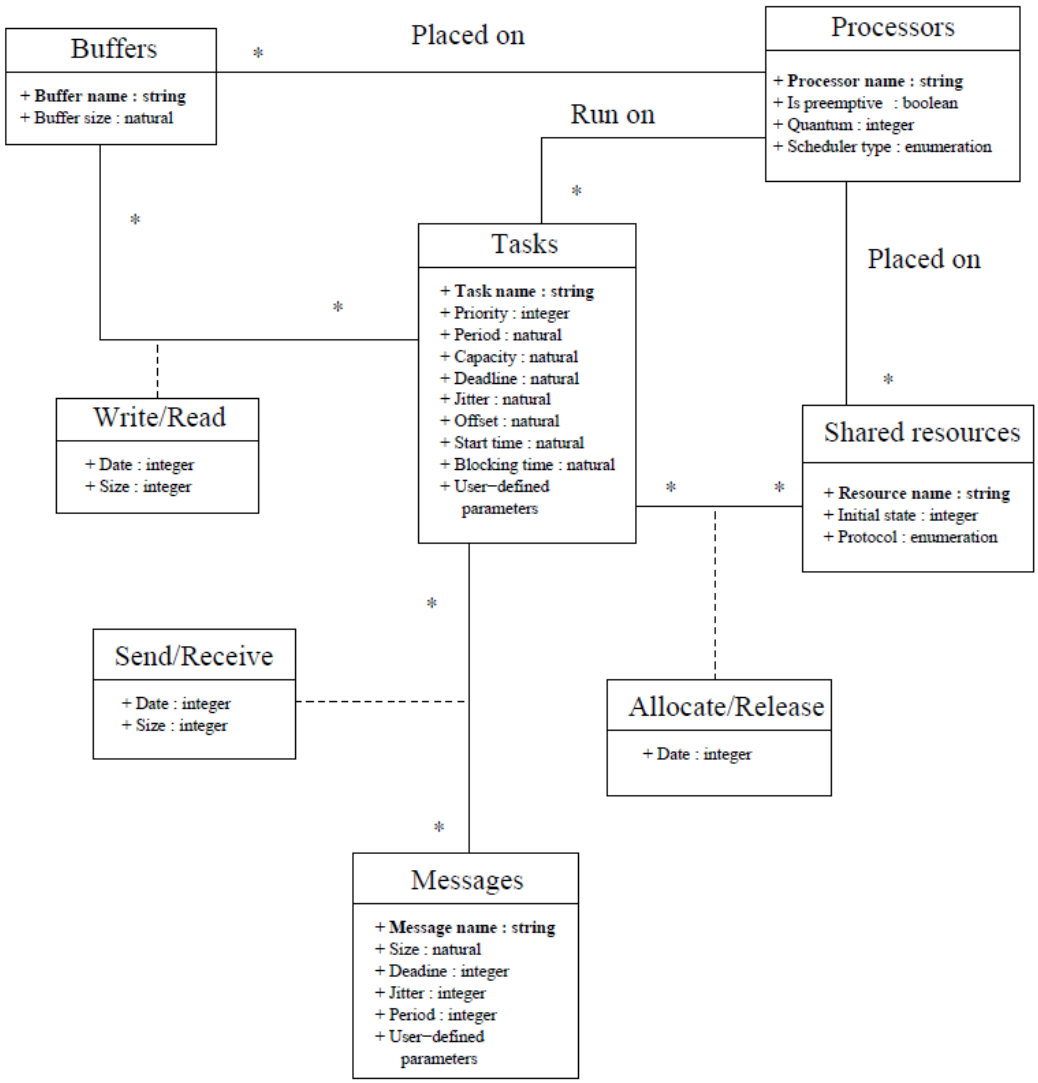
Το Cheddar είναι ένα πλαίσιο εργασίας σε Ada που παρέχει εργαλεία για τον έλεγχο αν μια εφαρμογή πραγματικού χρόνου ικανοποιεί τις χρονικές της προθεσμίες. Το πλαίσιο εργασίας βασίζεται στη θεωρία χρονοπρογραμματισμού πραγματικού χρόνου και είναι γραμμένο κυρίως για εκπαιδευτικούς σκοπούς. Στο Cheddar μία εφαρμογή ορίζεται σαν ένα σύνολο επεξεργαστών, εργασιών, απομονωτών, μοιραζόμενους πόρους και μηνύματα. Παρέχει ελέγχους χρονοπρογραμματισμού για μονοεπεξεργαστικά, πολυεπεξεργαστικά ή κατανεμημένα συστήματα. Το πλαίσιο εργασίας είναι ανοικτό και έχει σχεδιαστεί για να συνδέεται εύκολα με εργαλεία CASE όπως συντάκτες, σχεδιαστικά εργαλεία, προσομοιωτές και άλλα.

Οι δοκιμές εφικτότητας του Cheddar εστιάζουν σε συστήματα που δεν έχουν μελετηθεί εκτενώς όπως συστήματα με μοιραζόμενους απομονωτές ή εξαρτήσεις έργων. Μαζί με το πρόγραμμα διανέμεται και η βιβλιογραφία που περιγράφει πως υπολογίζονται οι δοκιμές εφικτότητας. Κάθε αποτέλεσμα που υπολογίζεται από το Cheddar εμφανίζεται με αναφορά στη μαθηματική έκφραση που χρησιμοποιήθηκε για τον υπολογισμό της. Έτσι μπορεί να χρησιμοποιηθεί από όσους θέλουν να κατανοήσουν τα θεμελιώδη στοιχεία της θεωρίας χρονοπρογραμματισμού πραγματικού χρόνου τόσο πρακτικά όσο και θεωρητικά.

Είναι ένα ανοικτό μεταφερτό και εύκολο στη χρήση πλαίσιο εργασίας. Αν και είναι object oriented έχει απλή διεπαφή χρήστη. Όλα τα δεδομένα που στέλνονται ή παράγονται από το πλαίσιο εργασίας είναι μορφής XML. Για ευκολία συντήρησης είναι γραμμένο σε Ada. Τρέχει σε όλα τα λειτουργικά συστήματα αλλά και κάθε πλατφόρμα που υποστηρίζει Gnat/Gtk Ada. Διανέμεται με άδεια GNU GPL.

Τέλος είναι ευέλικτο. Επειδή δοκιμές χρονοπρογραμματισμού υπάρχουν μόνο για λίγους καλά μελετημένους αλγόριθμους και σειρές έργων ο προσομοιωτής είναι αρκετά ευέλικτος να προσομοιώνει συστήματα με ειδικές χρονικές συμπεριφορές. Για την επέκταση του πλαισίου εργασίας χρησιμοποιεί μία γλώσσα ada-like. Οι επεκτάσεις των χρηστών που προγραμματίζονται σ' αυτή τη γλώσσα δεν μεταγλωττίζονται. Διερμηνεύονται από τον προσομοιωτή την ώρα της προσομοίωσης. Αυτό επιτρέπει στο σχεδιαστή να γράψει και να εκτελέσει εύκολα και γρήγορα ένα νέο χαρακτηριστικό χρονοπρογραμματισμού χωρίς βαθειά γνώση του σχεδιασμού του πλαισίου εργασίας και της γλώσσας Ada.

Δοκιμές και υπηρεσίες προσομοίωσης που παρέχονται από το cheddar.

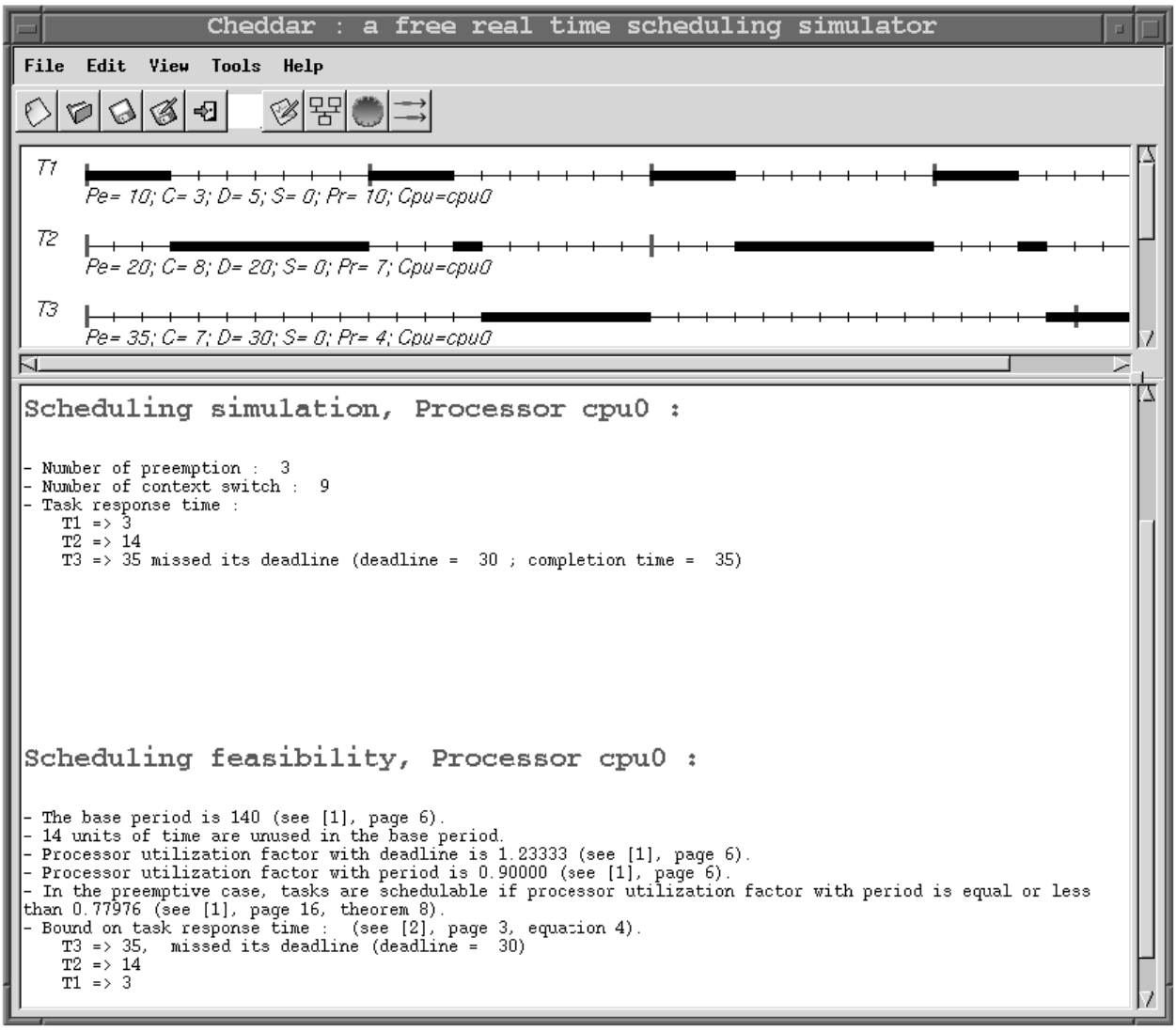


Σχήμα 3.18 Η δομή μιας εφαρμογής στο Cheddar**.**

Στο Cheddar μια εφαρμογή ορίζεται από ένα σύνολο επεξεργαστών, απομονωτών, μοιραζόμενων πόρων, μηνυμάτων και εργασιών. Η δομή φαίνεται στο σχήμα 3.18. Στο απλούστερο μοντέλο εργασιών κάθε εργασία εκτελείται περιοδικά. Αυτά τα περιοδικά έργα ορίζονται από τρεις παραμέτρους: το χρόνο εκτέλεσης Ci, την προθεσμία Di και την περίοδο Pi. Η περίοδος είναι το χρονικό διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών ενεργοποιήσεων του έργου i. Κάθε φορά που το έργο i ενεργοποιείται θα πρέπει να χρησιμοποιήσει τον επεξεργαστή για χρόνο Ci και να έχει ολοκληρώσει την εκτέλεσή του πριν την πάροδο χρόνου Di.

Για ένα σύνολο έργων δύο είδη ανάλυσης μπορούν να γίνουν: δοκιμές εφικτότητας και προσομοίωση του χρονοπρογραμματισμού.

Η προσομοίωση χρονοπρογραμματισμού αποτελείται από τη μελέτη κάθε χρονική στιγμή σε ποιο έργο έχει αποδοθεί για χρήση ο επεξεργαστής. Έλεγχος αν οι εργασίες προλαβαίνουν τις προθεσμίες τους μπορεί να γίνει από την ανάλυση του υπολογισθέντος χρονοπρογραμματισμού. Το σχήμα 3.19 απεικονίζει ένα σύνολο από 3 περιοδικά έργα τα τ_1, τ_2, τ_3 με περιόδους 10, 20, 35 και χρόνο εκτέλεσης 3, 8, 7 και προθεσμίες 5, 20, 30. Στο πάνω μέρος του παραθύρου φαίνεται η προσομοίωση του χρονοπρογραμματισμού του συνόλου των έργων. Αυτά έχουν χρονοπρογραμματιστεί με τον αλγόριθμο RM που είναι προεκτοπιστικός.



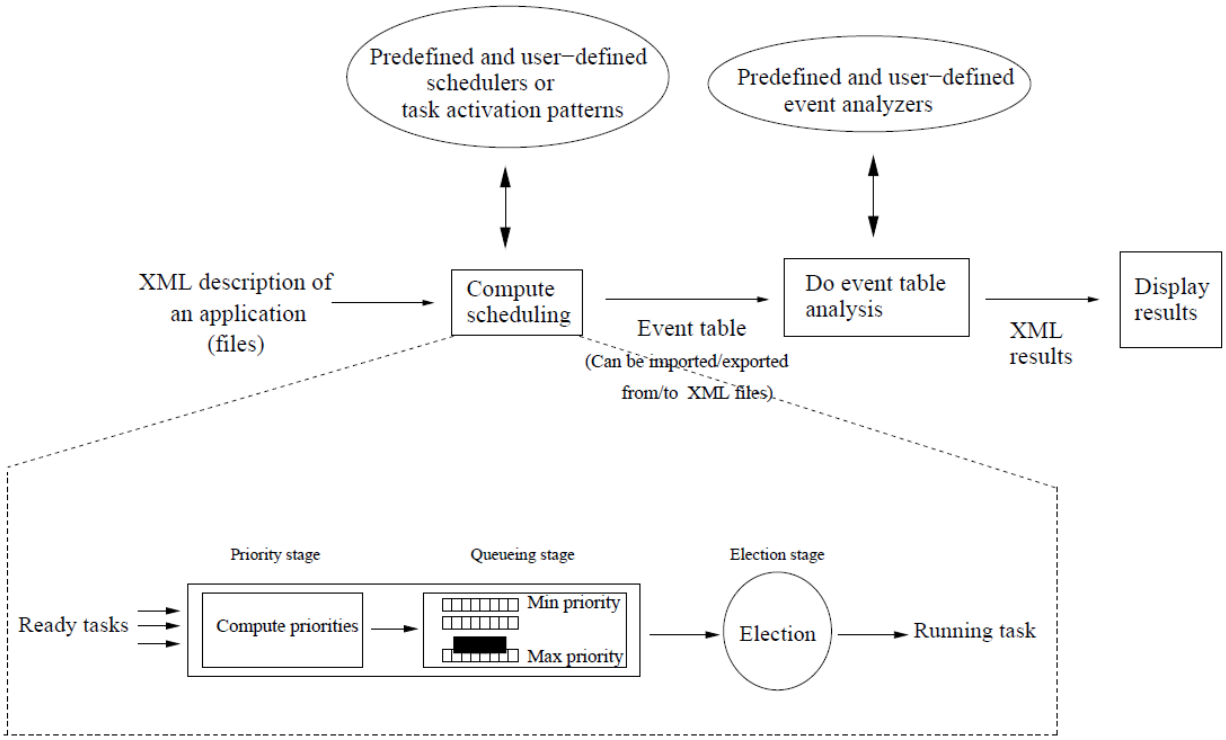
Σχήμα 3.19 Η οθόνη αποτελεσμάτων στο Cheddar.

Το cheddar έχει υλοποιημένους τους πιο συνηθισμένους χρονοπρογραμματιστές όπως RM, EDF, DM, LLF και POSIX χρονοπρογραμματιστές με πολιτικές ουράς SCHED_FIFO, SCHED_RR και SCHED_OTHER. Μετά τον χρονοπρογραμματισμό μπορεί να εξαχθούν από τον προσομοιωτή πληροφορίες όπως μέσος-χειρότερος-καλύτερος χρόνος απόκρισης, μέσος χρόνος αναμονής, καλύτερος-χειρότερος χρόνος αποκλεισμού, αριθμός προεκτοπίσεων, αριθμός εναλλαγών περιβάλλοντος, χρήση απομονωτών, συνολική χρονική καθυστέρηση μηνύματος επικοινωνίας από άκρη σε άκρη και άλλα.

Για ένα δοθέν σύνολο από εργασίες εάν η προσομοίωση χρονοπρογραμματισμού είναι πολύ μεγάλη για να υπολογιστεί μπορούν να εφαρμοστούν οι δοκιμές εφικτότητας. Υπάρχουν διάφορα είδη δοκιμών εφικτότητας. Δοκιμές που βασίζονται στη χρήση του επεξεργαστή, δοκιμές που υπολογίζουν το χρόνο απόκρισης ενός έργου και έχουν σχεδιαστεί για να ελέγχουν τις προθεσμίες και δοκιμές χρήσης απομονωτών που έχουν σχεδιαστεί να ελέγχουν την υπερχείλιση των απομονωτών. Ένας μεγάλος αριθμός από αποτελέσματα υπάρχει για χρονοπρογραμματισμό προεκτοπιστικό ή όχι αλλά ακόμη και για πιο πολύπλοκα σύνολα έργων με μοιραζόμενους πόρους και πρωτόκολλα PIP και PCP ή έργα με εξαρτήσεις προτεραιότητας. Ο σχεδιαστής συστημάτων θα βρει μερικές επεκτάσεις υλοποιημένες στο Cheddar. Επειδή ο αριθμός των δοκιμών είναι μεγάλος κάθε αποτέλεσμα που παρουσιάζεται έχει μία βιβλιογραφική αναφορά που σε αυτήν περιγράφεται ο τρόπος υπολογισμού του

Επεκτάσεις του cheddar.

Συνήθως οι δοκιμές χρονοπρογραμματισμού περιορίζονται σε λίγα μοντέλα έργων, κύρια περιοδικά, και μόνο για λίγους αλγορίθμους χρονοπρογραμματισμού. Όταν μία εφαρμογή δημιουργείται με μία συγκεκριμένη σειρά ενεργοποίησης των εργασιών ή ο χρονοπρογραμματισμός με ένα συγκεκριμένο αλγόριθμο πρέπει να ελεγχθεί, οι δοκιμές εφικτότητας δεν είναι απαραίτητες. Σ' αυτή την περίπτωση η μόνη λύση είναι η ανάλυση της προσομοίωσης του χρονοπρογραμματισμού. Το Cheddar επιτρέπει στο χρήστη να σχεδιάσει και εύκολα να χτίσει επεκτάσεις στο πλαίσιο εργασίας για να κάνει προσομοίωση με αλγορίθμους ορισμένους από το χρήστη ή συγκεκριμένες σειρές εργασιών. Στο σχήμα 3.20 δίνει μια ιδέα του τρόπου με τον οποίο υλοποιείται η μηχανή προσομοίωσης του πλαισίου εργασίας. Η εκτέλεση μιας προσομοίωσης στο Cheddar υλοποιείται σε τρία βήματα.



Σχήμα 3.20 Η οργάνωση του Cheddar.

Το πρώτο βήμα είναι οι υπολογισμοί του χρονοπρογραμματισμού για να αποφασίσουμε κάθε χρονική στιγμή ποιο συμβάν θα γίνεται. Συμβάντα μπορεί να είναι η αποδέσμευση ενός μοιραζόμενου πόρου, το διάβασμα ή το γράψιμο ενός απομονωτή, η λήψη ή η αποστολή μηνυμάτων και φυσικά η εκτέλεση ενός έργου μία δεδομένη χρονική στιγμή. Στο τέλος αυτού του βήματος δημιουργείται ένας πίνακας που έχει όλα τα δημιουργηθέντα συμβάντα. Ο πίνακας συμβάντων χτίζεται στην XML περιγραφή της εφαρμογής του χρονοπρογραμματιστή και των σειρών ενεργοποίησης των εργασιών. Συνήθως οι χρονοπρογραμματιστές και οι σειρές ενεργοποίησης συνόλου εργασιών έχουν οριστεί από πριν στο Cheddar αλλά ο χρήστης μπορεί να προσθέσει τους δικούς του.

Στο δεύτερο βήμα εκτελείται η ανάλυση των συμβάντων του πίνακα. Ο πίνακας σαρώνεται από τον αναλυτή συμβάντων για να βρει ιδιότητες του συστήματος που μελετάμε. Σ' αυτό το βήμα μερικές τυποποιημένες πληροφορίες μπορεί να εξαχθούν από προκαθορισμένους αναλυτές συμβάντων (χειρότερη-μέση-καλύτερη χρόνος Blocking , χαμένες προθεσμίες, κλπ.) αλλά οι χρήστες μπορεί να ορίσουν τους δικούς τους αναλυτές συμβάντων για ιδιότητες κατά περίπτωση (για παράδειγμα περιορισμοί συγχρονισμού μεταξύ δύο εργασιών, σειρά προσπέλασης ενός μοιραζόμενου πόρου). Τα αποτελέσμα-

τα που παράγονται σε αυτό το βήμα είναι μορφής XML και μπορούν να εξαχθούν σε άλλα προγράμματα.

Το τελευταίο βήμα είναι η παρουσίαση των αποτελεσμάτων στο κύριο παράθυρο του cheddar, όπως φαίνεται και στο σχήμα 3.19.

Ορισμός έργου.

Τώρα ας δούμε πως προστίθεται στο πλαίσιο εργασίας ένα νέο έργο και σειρές έργων από τον χρήστη. Βασικά όλα τα έργα αποθηκεύονται σε ένα διάνυσμα που λέγεται διάνυσμα TCB. Ο χρονοπρογραμματιστής πρέπει να βρει ένα έτοιμο για εκτέλεση έργο από αυτό το διάνυσμα. Για να το πετύχει αυτό οι χρονοπρογραμματιστές χτίζονται σε διοχέτευση τριών σταδίων (σχήμα 2.20).

Στάδιο προτεραιότητας : για κάθε έτοιμο έργο υπολογίζεται μία προτεραιότητα.

Στάδιο ουράς : τα έτοιμα έργα εισάγονται στις αντίστοιχες ουρές. Υπάρχει μια ουρά ανά επίπεδο προτεραιότητας. Κάθε ουρά περιέχει όλα τα έργα που έχουν την ίδια προτεραιότητα. Οι ουρές διαχειρίζονται σαν POSIX ουρές χρονοπρογραμματισμού. Εάν μία χρονοθυρίδα αποδίδεται από το χρονοπρογραμματιστή στην ουρά ο αλγόριθμός είναι ο SCHED_RR [19]. Διαφορετικά εφαρμόζεται ο SECHD_FIFO.

Στάδιο επιλογής : Ο χρονοπρογραμματιστής αναζητά τη μη άδεια ουρά με την υψηλότερη προτεραιότητα και αποδίδει τον επεξεργαστή σο έργο που βρίσκεται στην κεφαλή της ουράς. Το επιλεγέν έργο χρησιμοποιεί τον επεξεργαστή για μία χρονοθυρίδα αν ο αλγόριθμός είναι προεκτοπιστικός ή μέχρι να τελειώσει αν δεν είναι προεκτοπιστικός.

Ο ορισμός ενός νέου χρονοπρογραμματιστή είναι απλά η επιβάρυνση μερικών σταδίων. Οι χονοδρομολογητές που ορίζονται από το χρήστη αποθηκεύονται με την XML περιγραφή του αρχείου εφαρμογής και οργανώνονται σε τμήματα. Κάθε τμήμα δίνει τις οδηγίες που πρέπει στη μηχανή προσομοίωσης κατά την προσομοίωση. Ένας χρονοπρογραμματιστής αποτελείται από το τμήμα start, το τμήμα priority, το τμήμα election και το τμήμα ενεργοποίησης του έργου.

Στο τμήμα start ο σχεδιαστής ορίζει τις μεταβλητές που χρειάζεται και βάζει τον κώδικα αρχικοποίησης. Υπάρχουν δυο οικογένειες μεταβλητών οι δυναμικές και οι στατικές. Οι στατικές μεταβλητές περιγράφουν την εφαρμογή που μελετάμε και ορίζονται από ετικέτες XML (σχήμα 3.21) στο αρχείο περιγραφής της εφαρμογής. Οι τιμές των στατικών μεταβλητών ποτέ δεν αλλάζουν κατά την προσομοίωση και διαβάζονται από το XML αρχείο. Οι δυναμικές μεταβλητές είναι δεδομένα που συλλέγονται από το πλαίσιο εργασίας κατά το χρόνο της προσομοίωσης. Δείχνουν την κατάσταση του έργου, των επεξεργαστών και άλλων στοιχείων της μελετώμενης εφαρμογής.

Όλες οι μεταβλητές που χρησιμοποιούνται σ' ένα χρονοπρογραμματιστή πρέπει να έχουν ένα τύπο. Το πλαίσιο εργασίας παρέχει δύο τύπους : ένα βαθμωτό τύπο(integer, double, byte) και διανύσματα. Ένα διάνυσμα είναι ένας τύπος που αποθηκεύει ένα βαθμωτό δεδομένο ανά έργο, μήνυμα, απομονωτή ή μοιραζόμενο πόρο. Λειτουργίες διανυσμάτων μπορούν να εκτελεστούν σ' αυτές τις μεταβλητές.

Το τμήμα priority περιέχει τον απαραίτητο κώδικα για να υπολογίζει τις προτεραιότητες των εργασιών. Ο κώδικας που δίνεται εδώ καλείται κάθε φορά που πρέπει να ληφθεί μια απόφαση χρονοπρογραμματισμού (κάθε χρονική θυρίδα για τους προεκτοπιστικούς και όταν ένα έργο ολοκληρώνεται για τους μη προεκτοπιστικούς).

Στο τμήμα election η μηχανή χρονοπρογραμματισμού αποφασίζει ποιο έτοιμο έργο θα πάρει τον επεξεργαστή στην επόμενη χρονοθυρίδα.

Τέλος το τμήμα activation περιγράφει πως ενεργοποιούνται τα έργα κατά τη διάρκεια της προσομοίωσης. Στο Cheddar τρία είδη προκαθορισμένων σειρών ενεργοποίησης

υπάρχουν: περιοδικά, απεριοδικά και κατά Poisson. Τα απεριοδικά έργα ενεργοποιούνται μία φορά και τα περιοδικά και τα Poisson αρκετές. Στην περίπτωση περιοδικών έργων δύο διαδοχικές ενεργοποιήσεις του έργου καθυστερούν κατά σταθερό χρόνο που λέγεται περίοδος. Στην περίπτωση των Poisson δύο διαδοχικές ενεργοποιήσεις του έργου καθυστερούν χρόνους που ακολουθούν την εκθετική κατανομή. Εάν είναι αναγκαίο ο σχεδιαστής μπορεί να ορίσει πιο πολύπλοκες σειρές ενεργοποίησης όπως σποραδικά έργα με τυχαία ενεργοποίηση ή καταιγισμός ενεργοποιήσεων.

```
start_section:
    partition_duration : array (tasks_range) of integer;
    dynamic_priority : array (tasks_range) of integer;
    number_of_partition : integer :=2;
    current : integer :=0;
    time_partition : integer :=0;
    -- The partition scheduling table
    --
    partition_duration(0):=2;
    partition_duration(1):=4;
    time_partition:=partition_duration(current);
priority_section:
    if time_partition=0
        then current:=(current+1)
            mod number_of_partition;
        time_partition:=partition_duration(current);
    end if;
    -- Choose the task with the highest priority
    -- owned by the active partition
    --
    for i in tasks_range loop
        if tasks.task_partition(i)=current
            then dynamic_priority(i):=tasks.priority(i);
            else dynamic_priority(i):=0;
              tasks.ready(i):=false;
        end if;
    end loop;
    time_partition:=time_partition-1;
election_section:
    return max_to_index(dynamic_priority);
```

Σχήμα 3.21 Η οργάνωση του Cheddar.

Τώρα ας δούμε κάποια παραδείγματα. Ο πιο απλός χρονοπρογραμματιστής είναι ο απλός μονοτονικός ρυθμός.

```
election_section:
    return min_to_index(period);
```

Αυτός δίνει τον επεξεργαστή στο έργο με τη μικρότερη περίοδο [7]. Η περίοδος είναι μία στατική μεταβλητή ορισμένη από το XML αρχείο της εφαρμογής που μοντελοποιεί τη μελέτη μας. Για υλοποίηση του χρονοπρογραμματιστή RM δεν υπολογίζονται προτεραιότητες και δεν απαιτούνται μεταβλητές. Έτσι ο σχεδιαστής δεν πρέπει να ορίσει τα τμήματα start και priority.

Το τμήμα election περιέχει μια πρόταση return για να ενημερώσει το πλαίσιο εργασίας ποιο έργο θα εκτελεστεί στην επόμενη χρονοθυρίδα. Το return χρησιμοποιεί τη συνάρτηση min_to_index. Αυτή η συνάρτηση εκτελεί διανυσματική λειτουργία. Σαρώνει το διάνυσμα TCB Για να βρει το έτοιμο έργο με την ελάχιστη τιμή για τη στατική μεταβλητή περίοδος.

Ψάχνοντας ιδιαιτερότητες κατά περίπτωση.

Κατά τον ίδιο τρόπο οι χρήστες μπορούν να ορίσουν νέους αναλυτές συμβάντων. Αυτοί είναι επίσης γραμμένοι σε μία γλώσσα Ada-like και διερμηνεύονται κατά την προσομοίωση.

Ο πίνακας συμβάντων που παράγεται από τον προσομοιωτή καταγράφει γεγονότα με την εκτέλεση του έργου και σχετίζεται με τα αντικείμενα που προσπελαύνει το έργο. Παραδείγματα συμβάντων που αποθηκεύονται σ' αυτό τον πίνακα μπορεί να είναι:

Συμβάντα που παράγονται όταν ένα έργο γίνεται έτοιμο για εκτέλεση (συμβάν task_activation) όταν ένα έργο αρχίζει ή τελειώνει (συμβάν start_of_task_capacity, end_ of_task_capacity).

Συμβάντα που παράγονται όταν ένα έργο διαβάζει ή γράφει δεδομένα από ή σε ένα απομονωτή (συμβάν read_from_buffer, write_to_buffer).

Συμβάντα που παράγονται όταν ένα έργο στέλνει ή λαμβάνει ένα μήνυμα (συμβάν receive_from_message, sed_to_message).

Συμβάντα που παράγονται όταν ένα έργο μπαίνει στην αναμονή για ένα απασχολημένο μοιραζόμενο πόρο (συμβάν allocate_resource, release_resource).

Κάθε συμβάν αποθηκεύεται μαζί με το χρόνο που συνέβη και την πληροφορία που σχετίζεται με αυτό καθ' εαυτό το συμβάν (πχ το όνομα του πόρου, του μηνύματος)

Ο πίνακας συμβάντων σαρώνεται σειριακά από τον αναλυτή συμβάντων. Οι αναλυτές που ορίζονται από το χρήστη αποτελούνται από αρκετά τμήματα όπως τα: start, gathering, analyse and display.

Το τμήμα start περιέχει τις δηώσεις και τις αρχικοποιήσεις των μεταβλητών.

Το τμήμα gathering περιέχει κώδικα που καλείται κάθε φορά που ανακαλείται ένα στοιχείο του πίνακα συμβάντων.

Τέλος το τμήμα display εκτελεί την ανάλυση των δεδομένων που αποθηκεύτηκαν προηγουμένως από το τμήμα gathering και εμφανίζει τα αποτελέσματα στο παράθυρο του Cheddar.

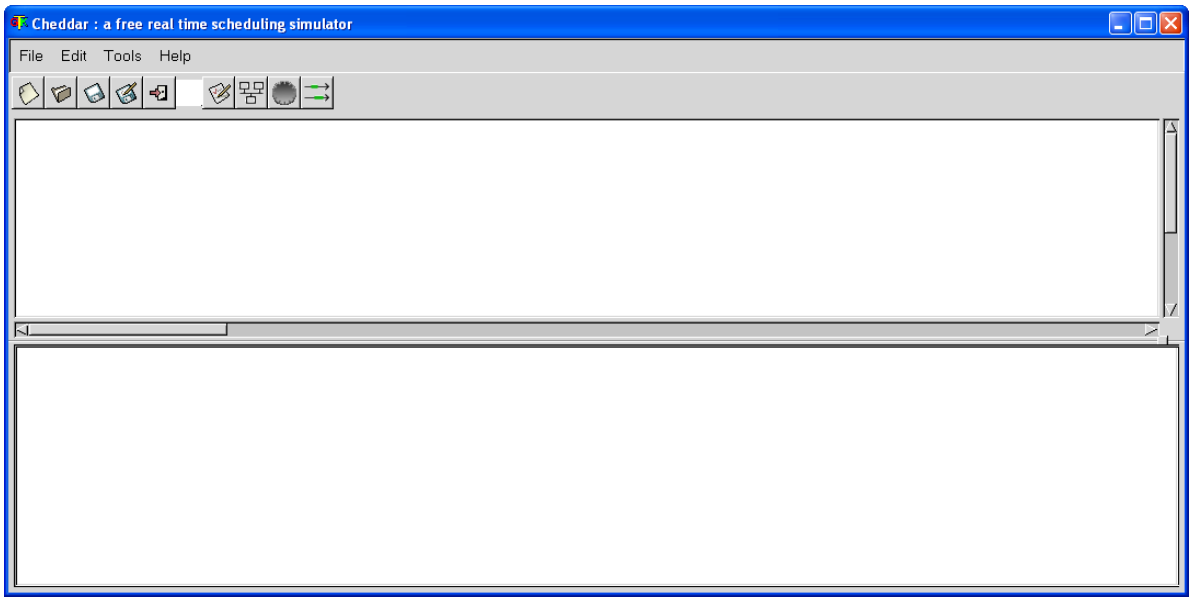
3.4 Παραδείγματα προσομοίωσης χρονοπρογραμματισμού με το Cheddar.

Έγινε προσπάθεια να βρεθούν όλοι οι προσομοιωτές και να γίνει από μία προσομοίωση στον καθένα δυστυχώς όμως προσπελάσιμος από το Διαδίκτυο ήταν μόνο ο

cheddar[39]. Σε αυτή την παράγραφο κάνουμε μία παρουσίαση του cheddar από την πλευρά του χρήστη και στη συνέχεια παρουσιάζουμε μερικές ασκήσεις παραδείγματα χρήσης του. Στο παράρτημα Ι έχουμε τις οθόνες από την προσομοίωση όλων των παραδειγμάτων που αναφέρονται σε αυτή την εργασία.

3.4.1 Εγκατάσταση και βασικές οδηγίες χρήσης.

Η εγκατάσταση του cheddar είναι απλή και γρήγορη. Λειτουργεί σε linux και windows. Εδώ περιγράφουμε τη χρήση του στα windows αντίστοιχη όμως είναι και η χρήση στο linux. Ένα σημαντικό στοιχείο είναι ότι η εγκατάσταση δεν τροποποιεί κανένα αρχείο των windows και η εγκατάσταση ολοκληρώνεται κυριολεκτικά σε πέντε λεπτά. Από την ιστοθέση http://beru.univ-brest.fr/~singhoff/cheddar/#Ref2 κατεβάζετε το αρχείο Cheddar-2.1-win32-bin.zip. και το αποσυμπιέζετε. Ο εξ' ορισμού φάκελος εγκατάστασης είναι ο Cheddar-2.1win32-bin. Στο φάκελο Cheddar-2.1win32-bin\docs βρίσκονται όλες οι εργασίες που χρησιμοποιεί το πρόγραμμα στην ανάλυση της προσομοίωσης και τη μελέτη της εφικτότητας του χρονοπρογραμματισμού. Το πρόγραμμα αρχίζει να εκτελείται με διπλό πάτημα στο αρχείο cheddar.exe. Η βασική οθόνη είναι αυτή του σχήματος 3.22.



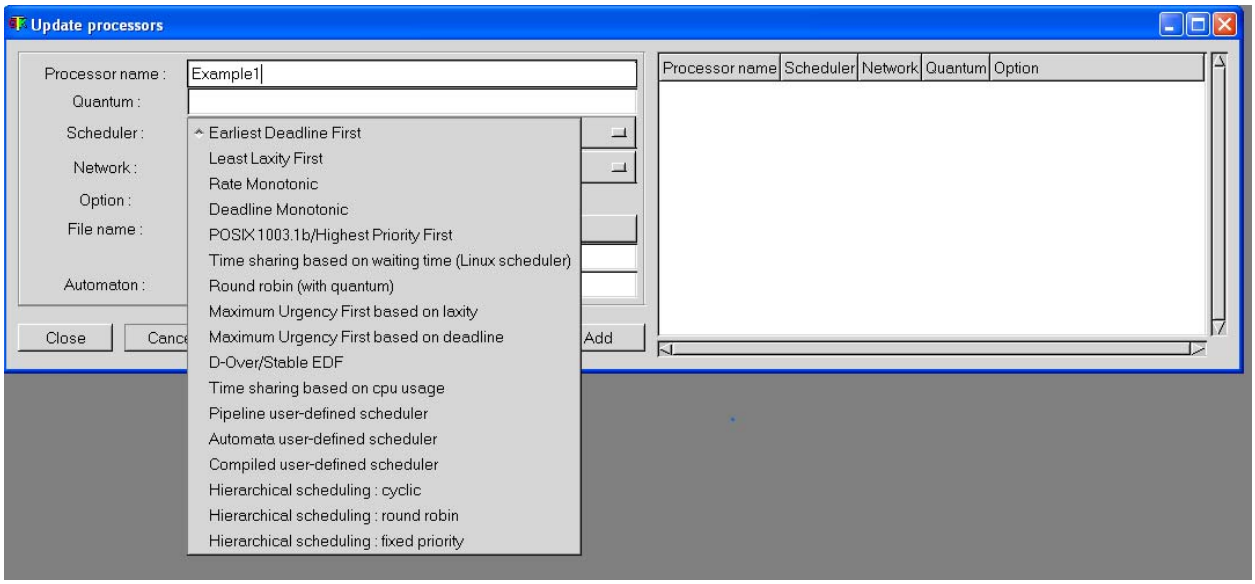
Σχήμα 3.22 Η βασική εικόνα του Cheddar.

Από το μενού File μπορείτε να δημιουργείστε, να ανοίξτε ή να αποθηκεύσετε μία προσομοίωση.

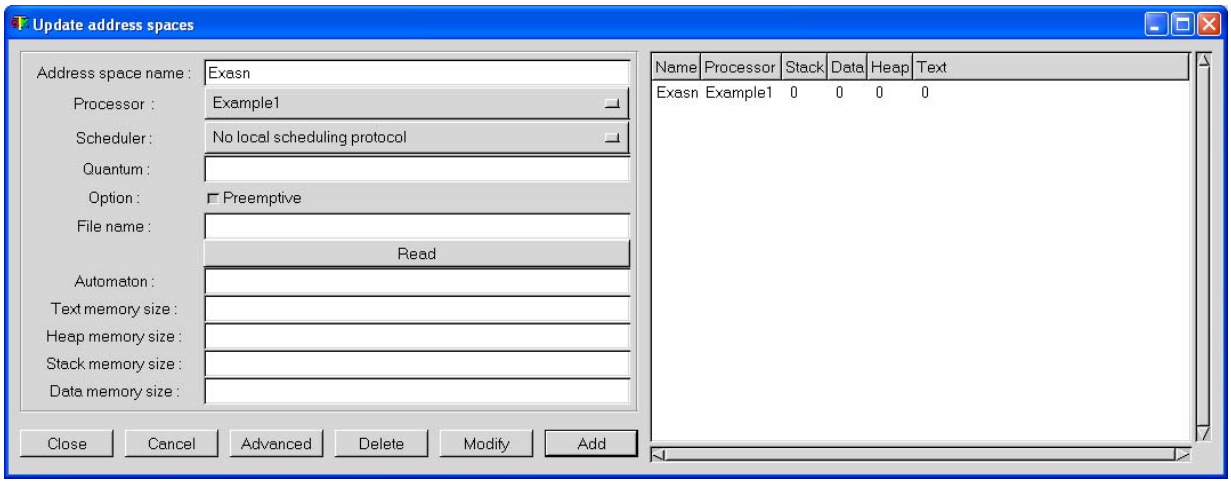
Στο μενού Help στην επιλογή scheduling references βρίσκονται αριθμημένες οι εργασίες που χρησιμοποιεί το πρόγραμμα για να μας ενημερώνει βάσει ποιού κριτηρίου ή ποιού μαθηματικού τύπου έγινε κάποια ενέργεια. Στο φάκελο Docs μπορείτε να βρείτε τις εργασίες αλλά δυστυχώς μόνο με ένα σύντομο όνομα. Αυτό στη αρχή κάνει επίπονη την αναζήτηση της εργασίας αναφοράς όταν θέλει κανείς να μελετήσει ή να αναλύσει περισσότερο κάποιο συμβάν.

Μία προσομοίωση ορίζεται σε τρία βήματα. Πρώτα στο μενού File>Update processors ορίζουμε τον επεξεργαστή που θα προσομοιώσει το σύστημά μας. Στη συνέχεια στο μενού File>Update address spaces ορίζουμε το χώρο μνήμης της προσομοίωσης – έχει νόημα όταν το σύστημα χρησιμοποιεί Buffers. Τέλος στο μενού File>Update Tasks ορίζουμε τις εργασίες που περιλαμβάνει το σύστημά μας και τα βασικά τους χαρακτηριστικά. Μία «ιδιοτροπία» του προγράμματος είναι πρώτα πατάμε Add για να γίνει μία κατα-

χώρηση και μετά close για να φύγουμε. Στα σχήματα 3.23, 3.24, 3.25 παρουσιάζουμε τις αντίστοιχες οθόνες.

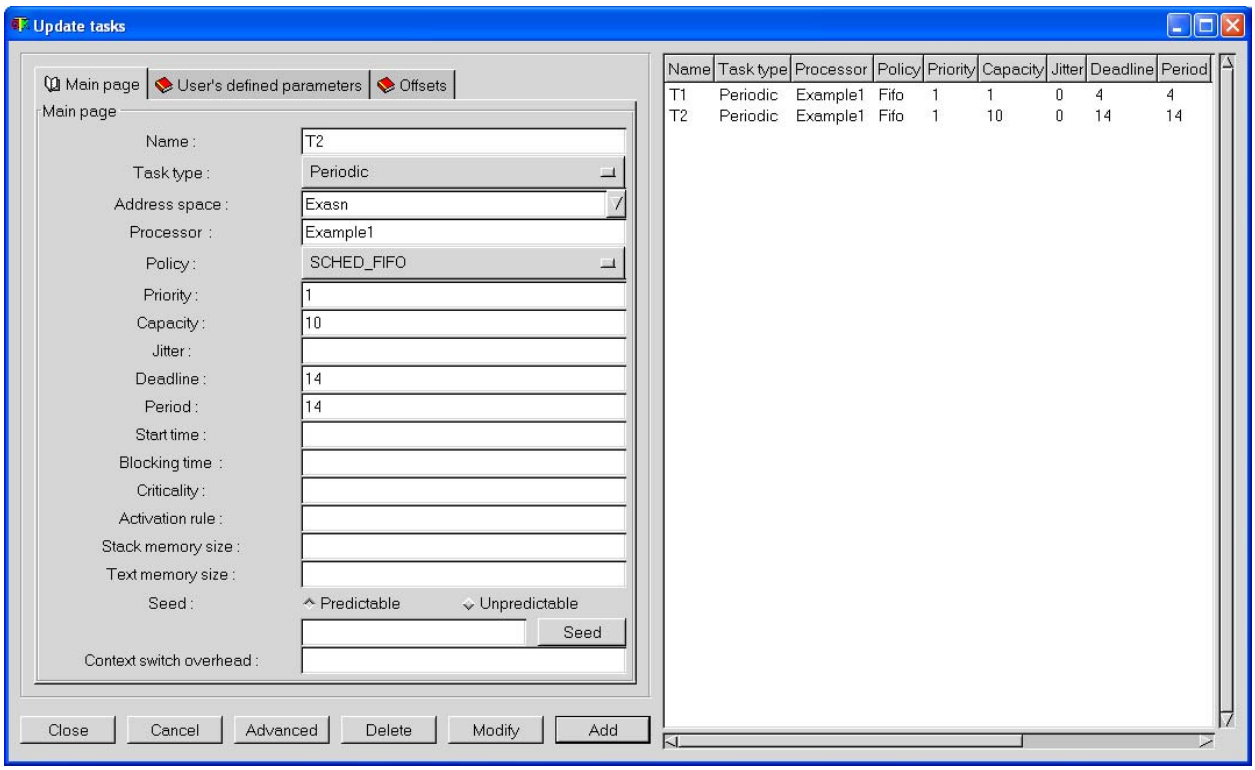


Σχήμα 3.23 Ορισμός processor.

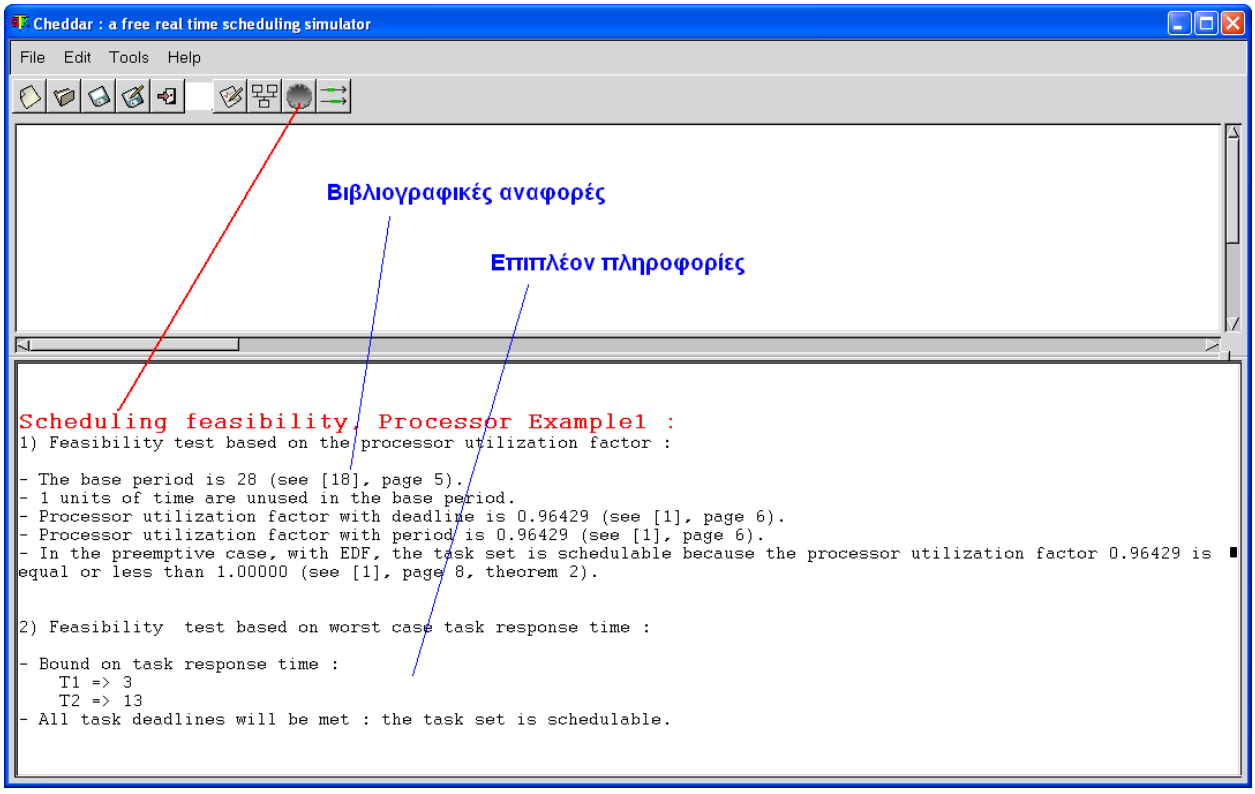


Σχήμα 3.24 Ορισμός του Address Space Name.

Μπορούμε να επιστρέφουμε και να αλλάζουμε τις παραμέτρους όταν είναι ανάγκη για να δούμε μια εναλλακτική επιλογή και να κάνουμε συγκρίσεις. Μπορούμε να κάνουμε είτε δοκιμή εφικτότητας του χρονοπρογραμματισμού πατώντας στο αντίστοιχο κουμπί, όπως στο σχήμα 3.26, και το πρόγραμμα μας ενημερώνει για τον χρονοπρογραμματισμό των εργασιών ή όχι.

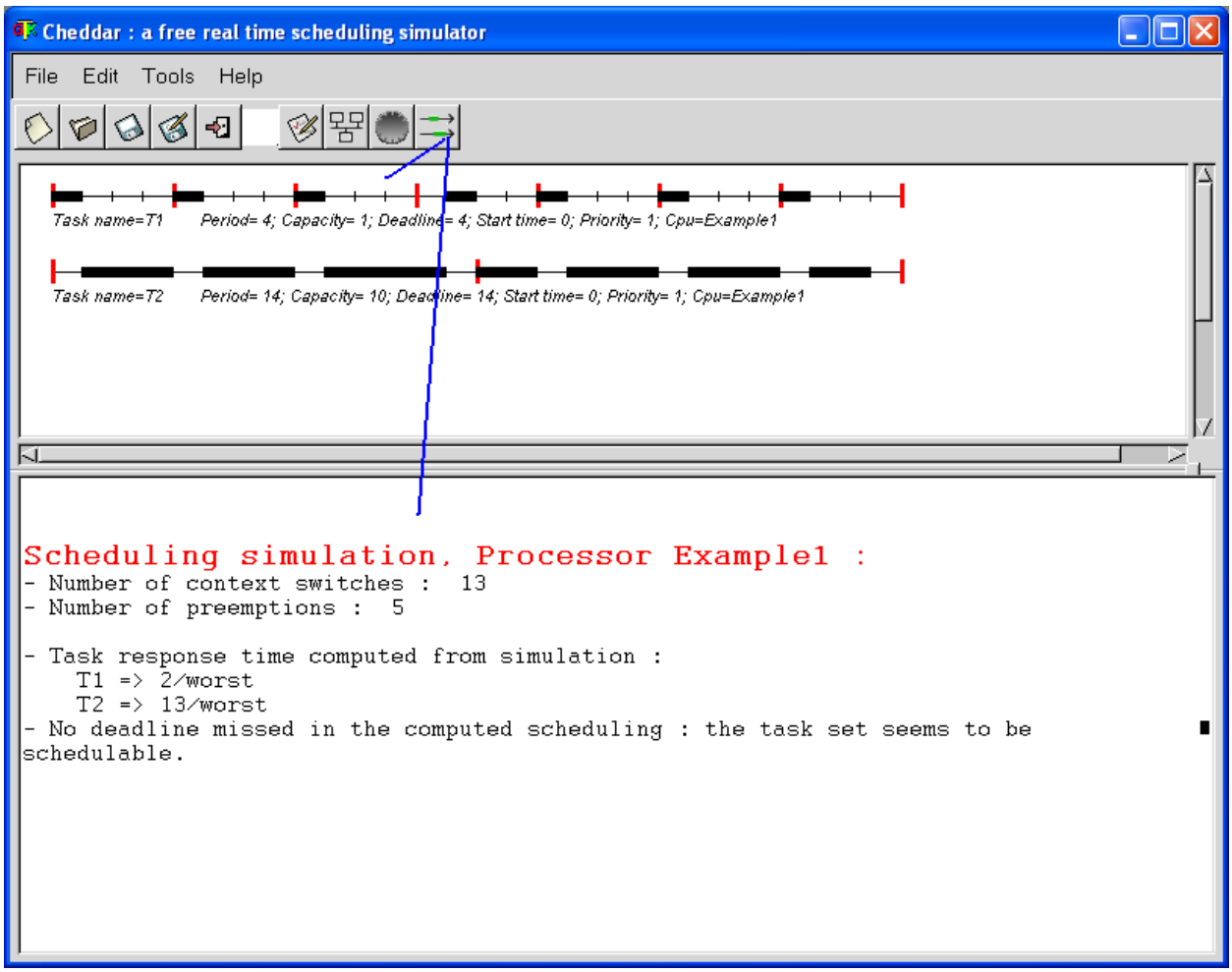


Σχήμα 3.25 Καταχώρηση έργων.



Σχήμα 3.26 Εκτέλεση μελέτης χρονοπρογραμματισμού.

Είτε να εκτελέσουμε την προσομοίωση για κάποιο χρονικό διάστημα (το πρόγραμμα προτείνει τη βασική περίοδο των εργασιών) αλλά μπορούμε να ορίσουμε όποιο άλλο διάστημα θέλουμε όπως φαίνεται στο σχήμα 3.27.

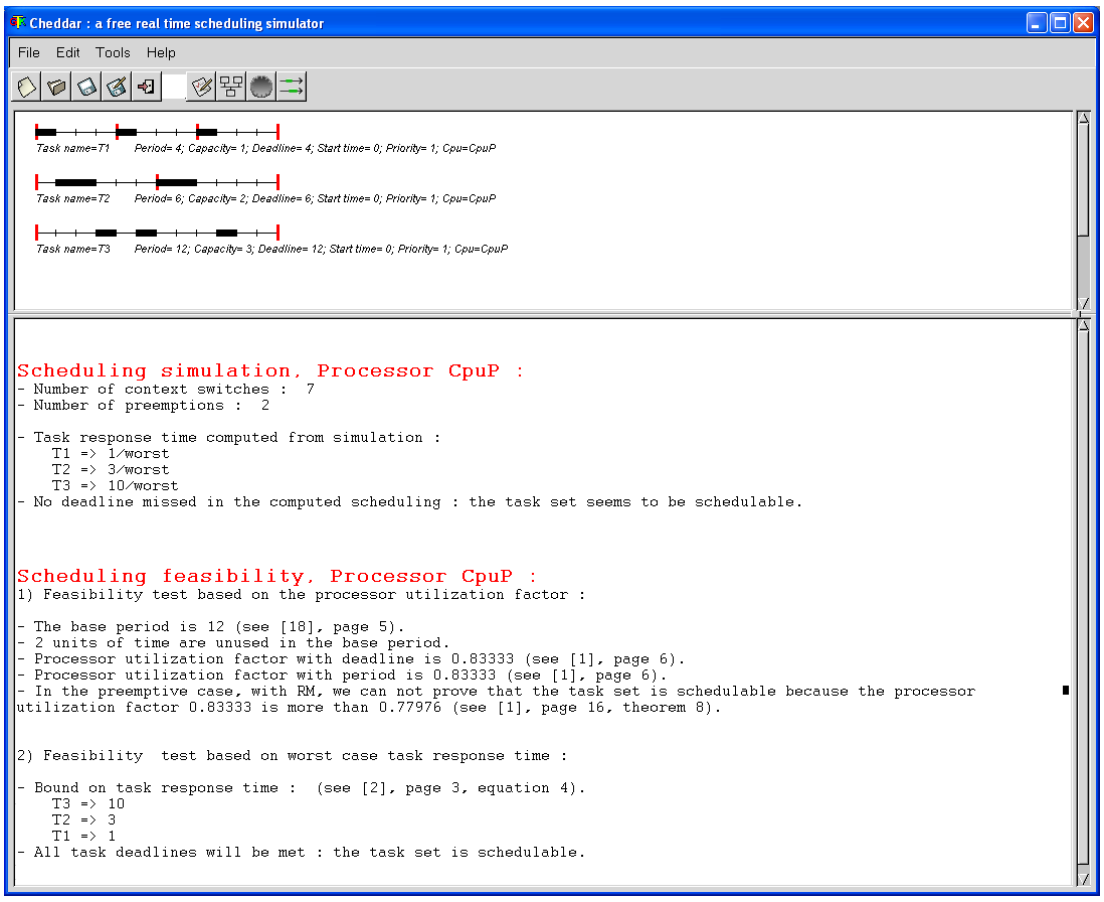


Σχήμα 3.27 Εκτέλεση προσομοίωσης.

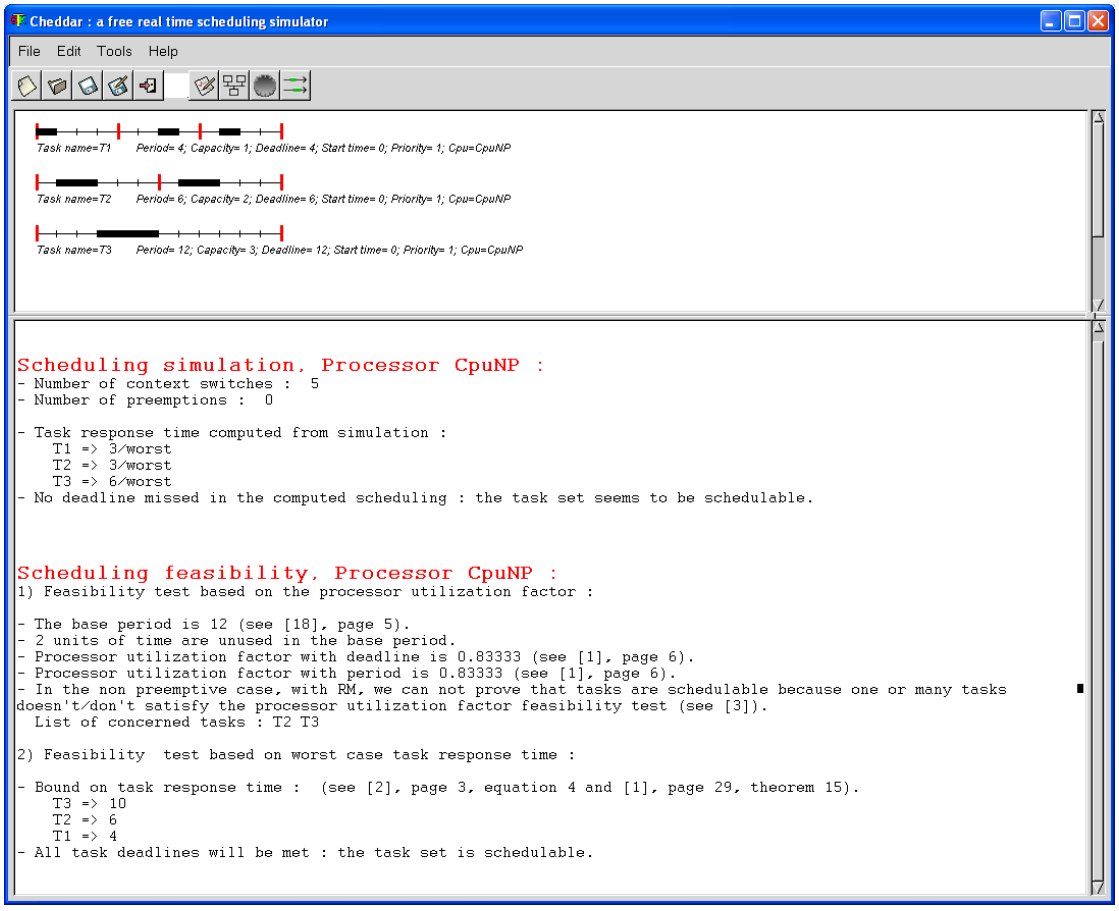
3.4.2 Άσκηση παράδειγμα.

Ένα σύστημα αποτελείται από τρεις εργασίες $\tau_1(0,1,4,4$, $\tau_2(0,2,6,6)$ και $\tau_3(0,3,12,12)$. Το σύστημα θα χρονοπρογραμματιστεί με τον αλγόριθμο RMS. Θέλουμε να μελετήσουμε πως θα εκτελούνται οι εργασίες αν ισχύει ο προεκτοπισμός και πως θα εκτελούνται οι εργασίες αν δεν ισχύει. Η RM-Preemptive φαίνεται στο σχήμα 3.28 και η RM-Non Preemptive στο σχήμα 3.29.

Μελετώντας τα διαγράμματα βλέπουμε ότι στο χρονοπρογραμματισμό CpuP (RMS preemptive) η σειρά ολοκλήρωσης των εργασιών είναι, σε μία βασική περίοδο, T1-T2-T1-T2-T1-T3. Στο χρονοπρογραμματισμό CpuNP (RMS Non-preemptive) η σειρά ολοκλήρωσης των εργασιών είναι, σε μία βασική περίοδο, T1-T2-T3-T1-T2-T1. Ενώ δεν έχει αλλάξει ο συντελεστής χρήσης του επεξεργαστή η διαδοχική εκτέλεση των έργων είναι διαφορετική. Ανάλογα του τι εργασία κάνει το σύστημα που προσομοιώνουμε αυτό μπορεί να είναι είτε αδιάφορο (πχ μία αποστολής ταινιών) είτε πολύ σημαντικό (μία εφαρμογή που προσομοιώνει τη διαδικασία παρασκευής ενός προϊόντος ανακατεύοντας υλικά με συγκεκριμένη δοσολογία και σειρά).

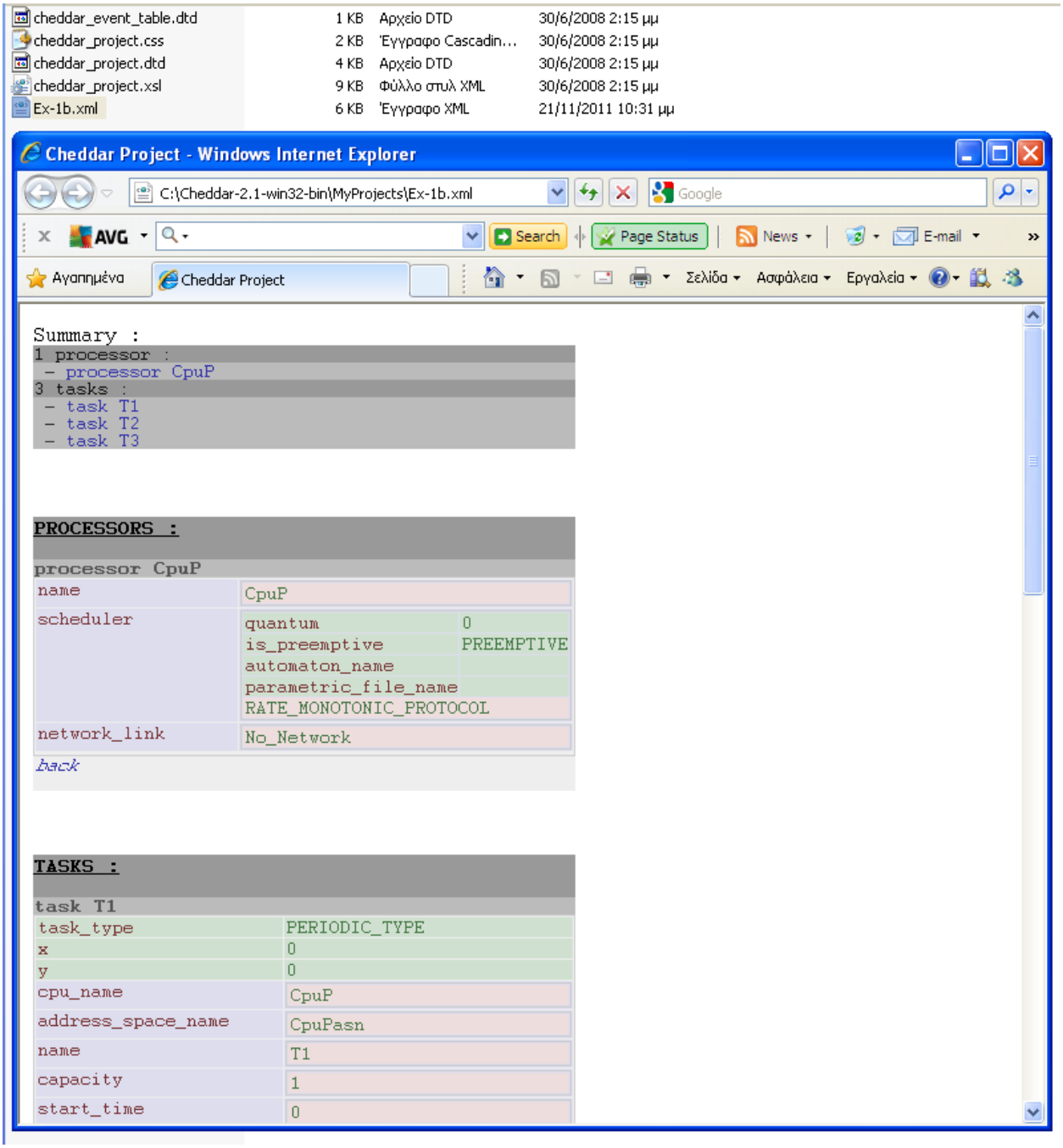


Σχήμα 3.28 Εκτέλεση προσομοίωσης RMS-Preemptive.



Σχήμα 3.29 Εκτέλεση προσομοίωσης RMS-No Preemptive.

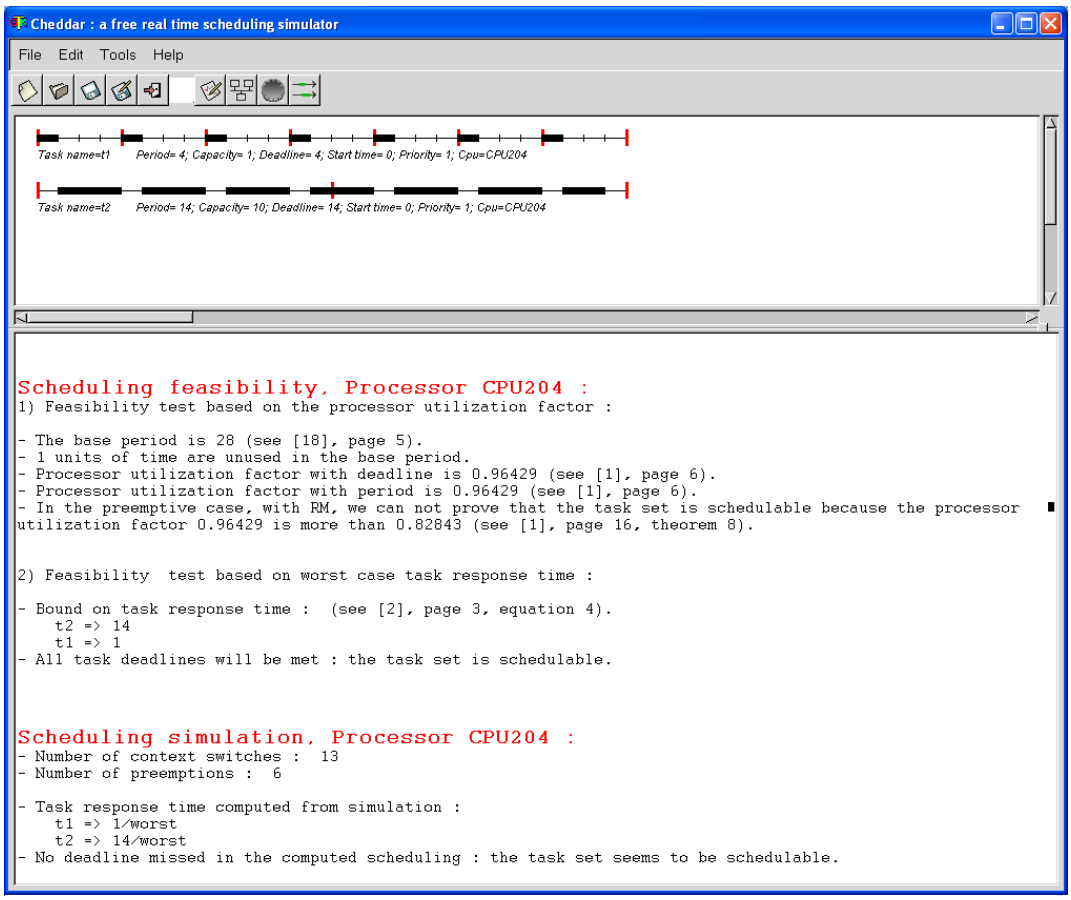
Μπορείτε να δείτε το σύστημα που περιγράψατε σε ένα πρόγραμμα περιήγησης όπως ο Internet Explorer. Δημιουργείστε ένα φάκελο με όνομα MyProjects και αντιγράψτε εκεί τα αρχεία cheddar_event_table.dtd, cheddar_project.css, cheddar-project.dtd. και cheddar-project.xsl. Τα αρχεία αυτά βρίσκονται στο φάκελο project-examples\xml του Cheddar. Αποθηκεύστε το αρχείο με τύπο xml στο φάκελο MyProjects. Τα *.xml αρχείο με διπλό πάτημα εμφανίζεται στον φυλλομετρητή σας, όπως στο σχήμα 3.30.



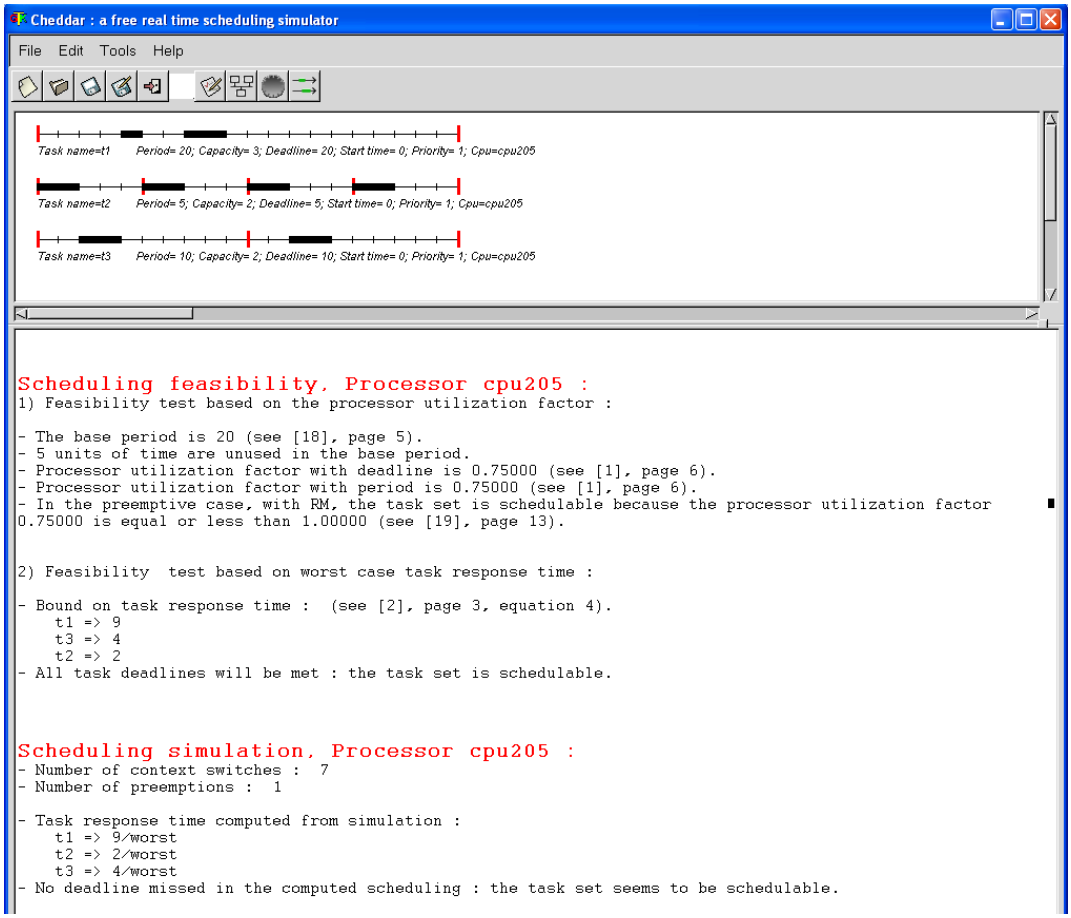
Σχήμα 3.30 Εμφάνιση του Xml αρχείου.

3.4.3 Τα παραδείγματα.

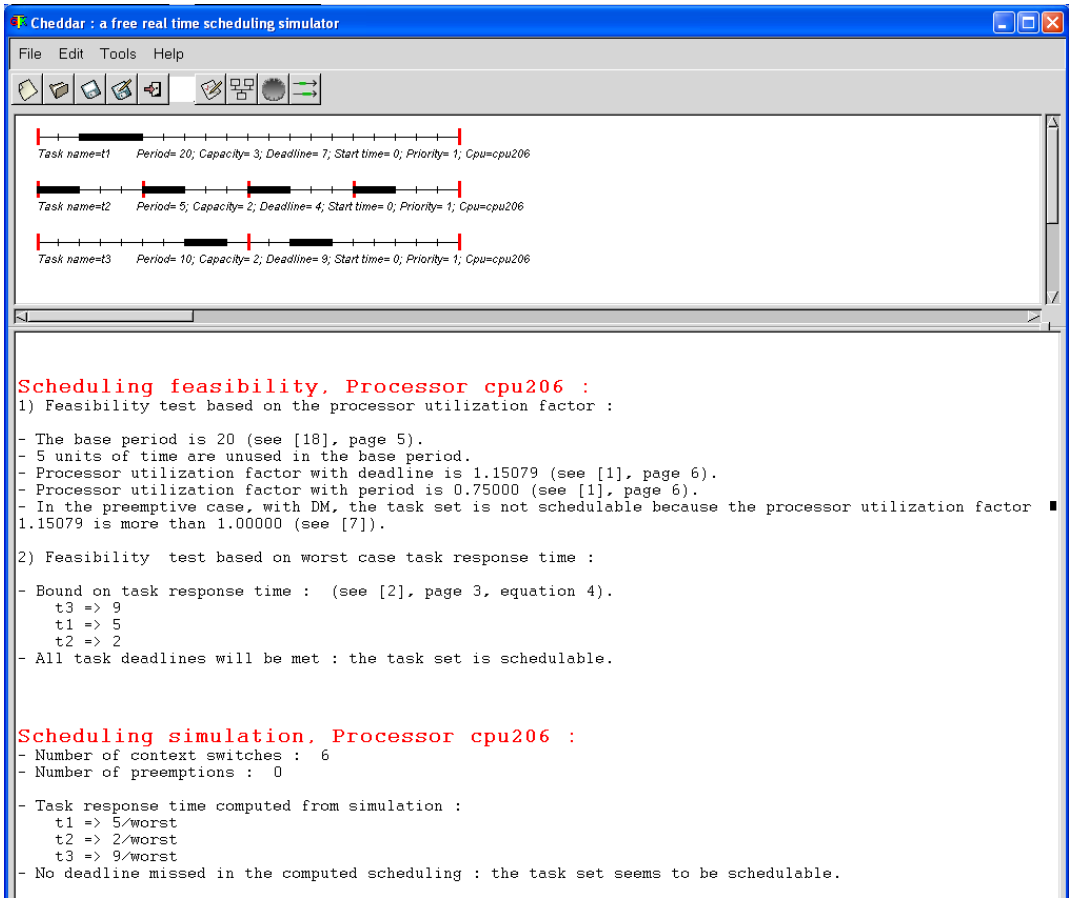
Τα παραδείγματα που αναφέρονται στο κεφάλαιο 3 τα εκτελέσαμε στο Cheddar. Τα αποτελέσματα της προσομοίωσης φαίνονται στα επόμενα σχήματα. Η λεζάντα του κάθε σχήματος παραπέμπει στο ποιο σύνολο έργων προσομοιώνει.



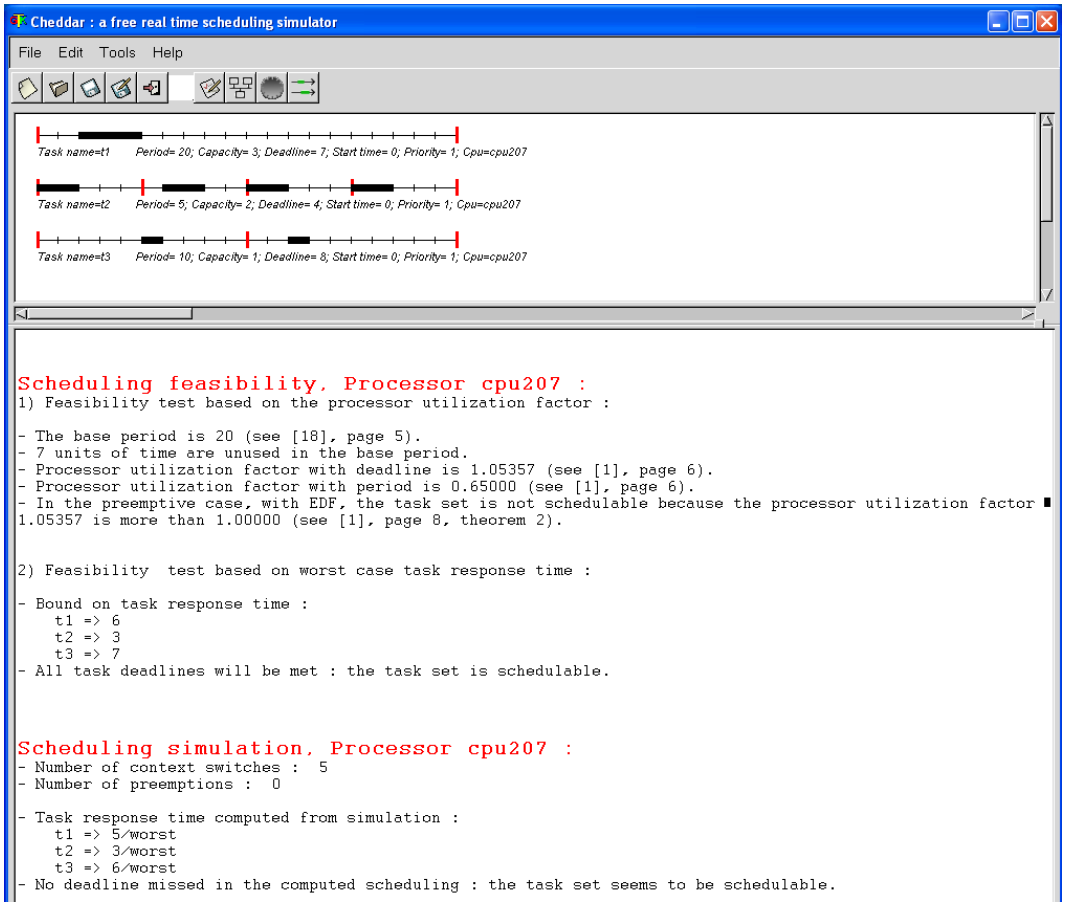
Σχήμα 3.31 Προσομοίωση σχήματος 3.4



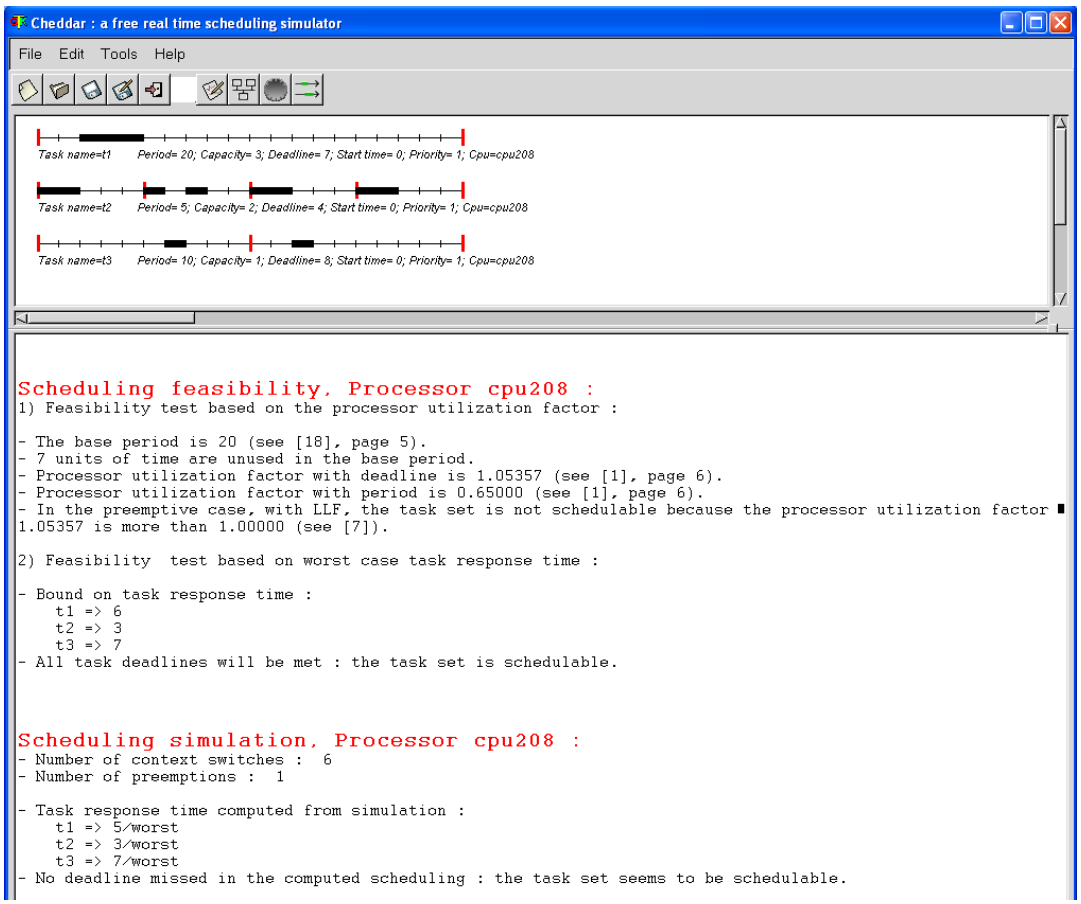
Σχήμα 3.32 Προσομοίωση σχήματος 3.5



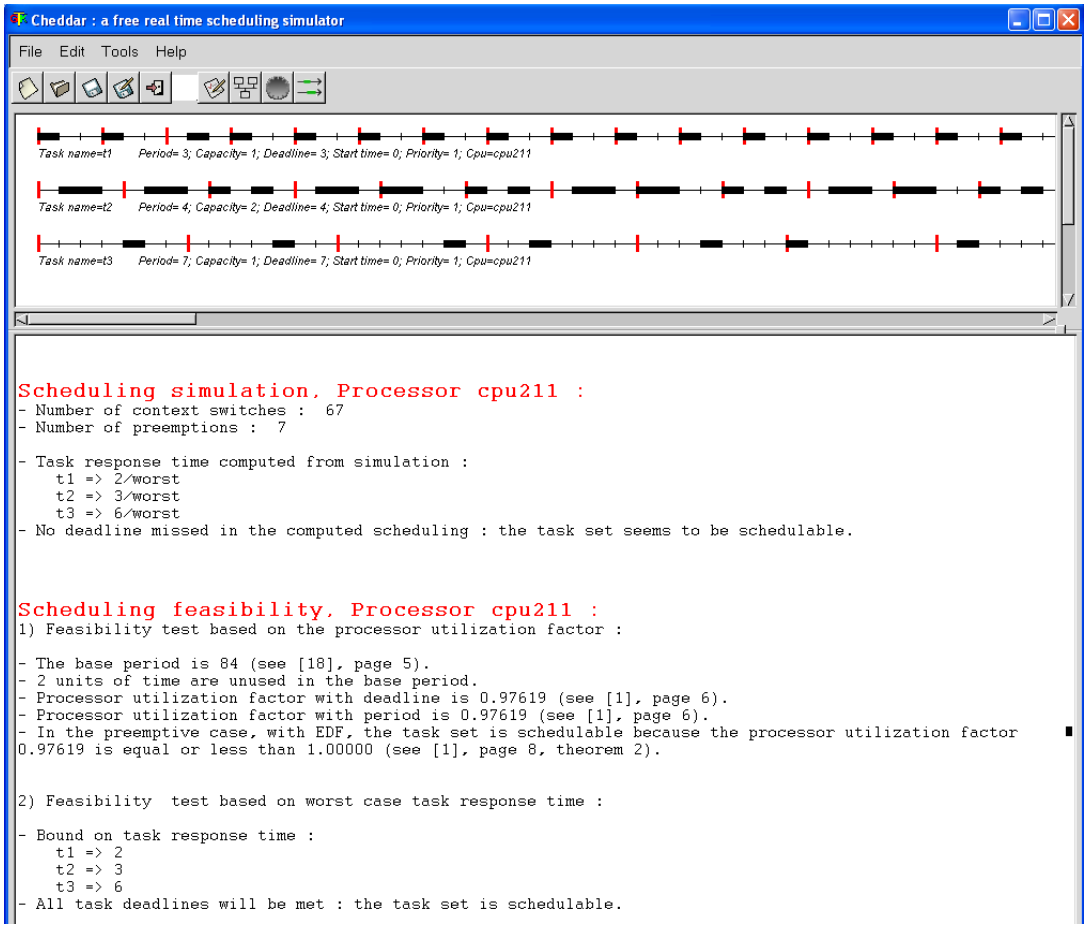
Σχήμα 3.33 Προσομοίωση σχήματος 3.6



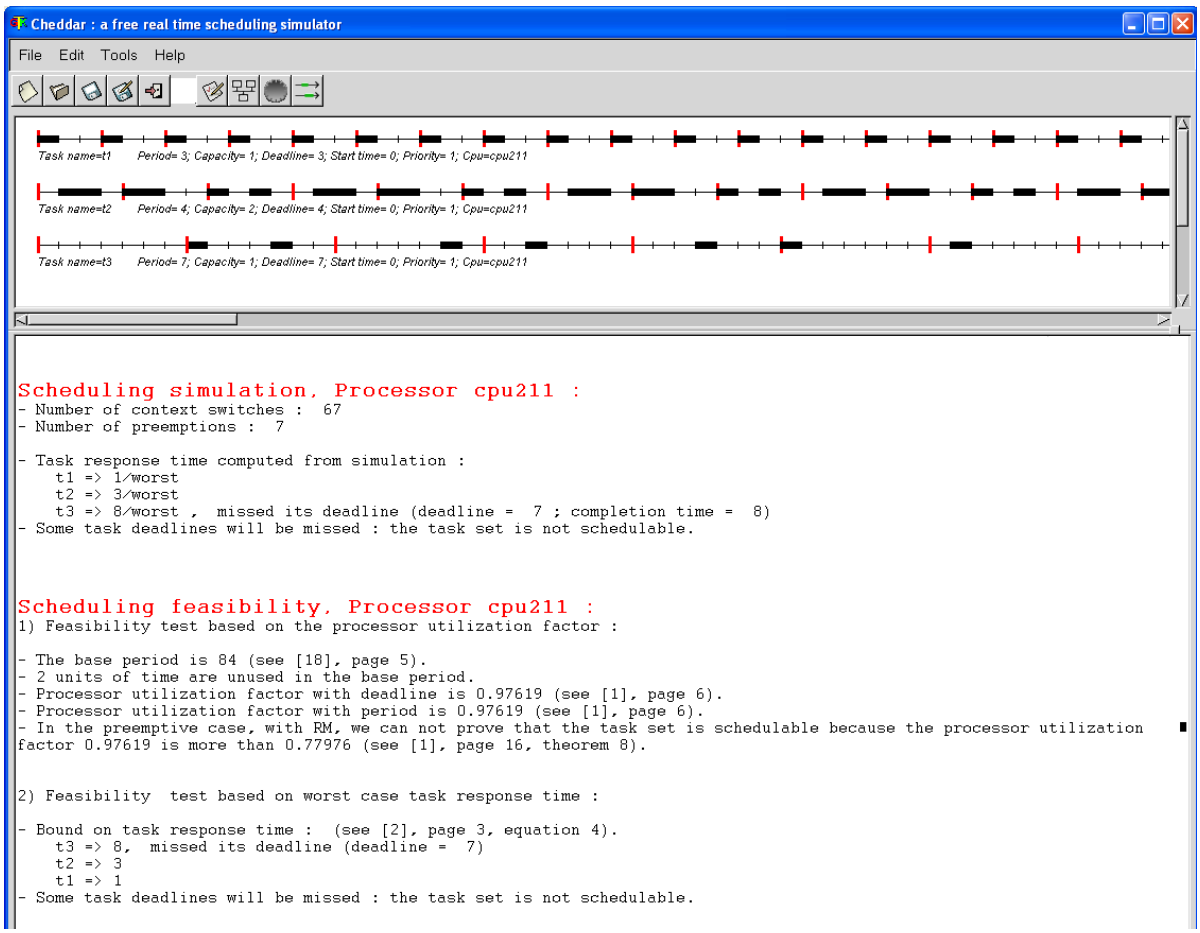
Σχήμα 3.34 Προσομοίωση σχήματος 3.7



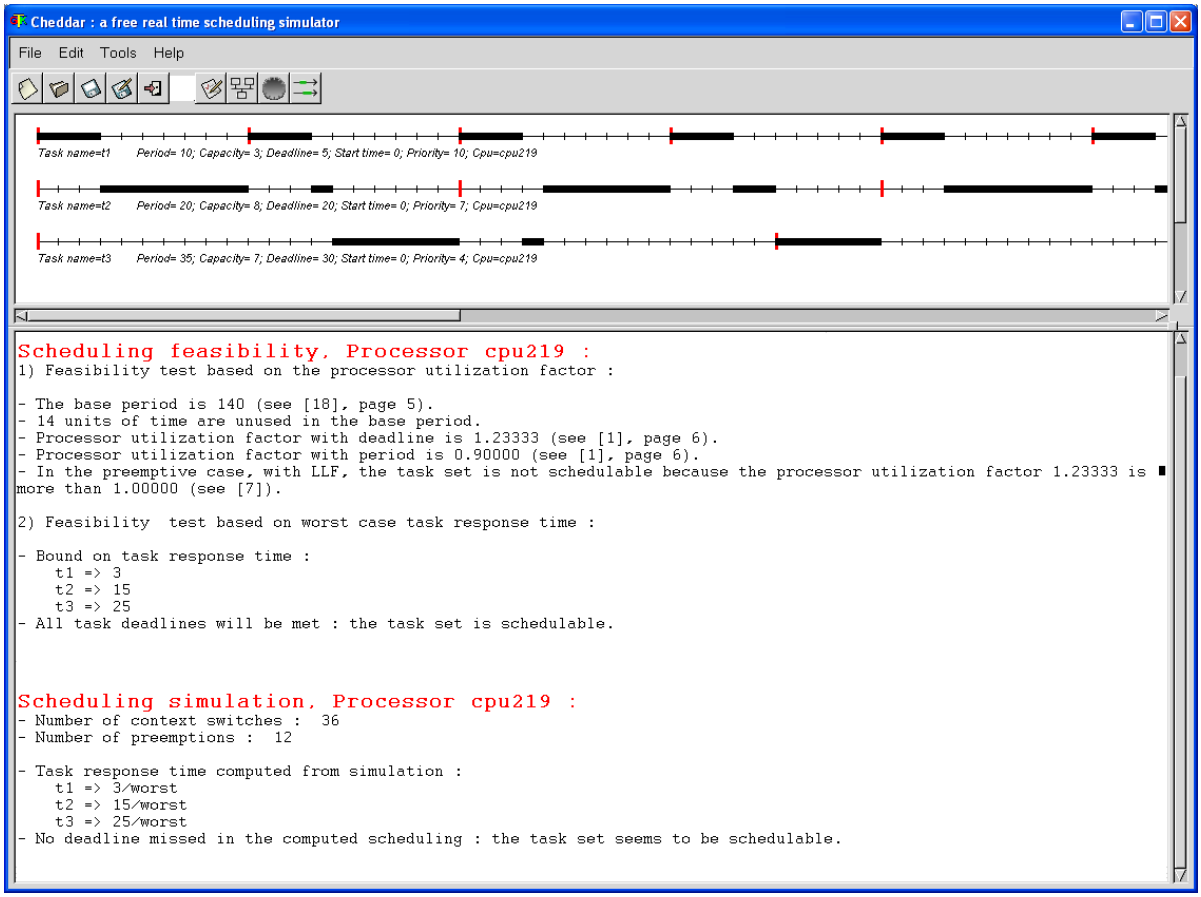
Σχήμα 3.35 Προσομοίωση σχήματος 3.8



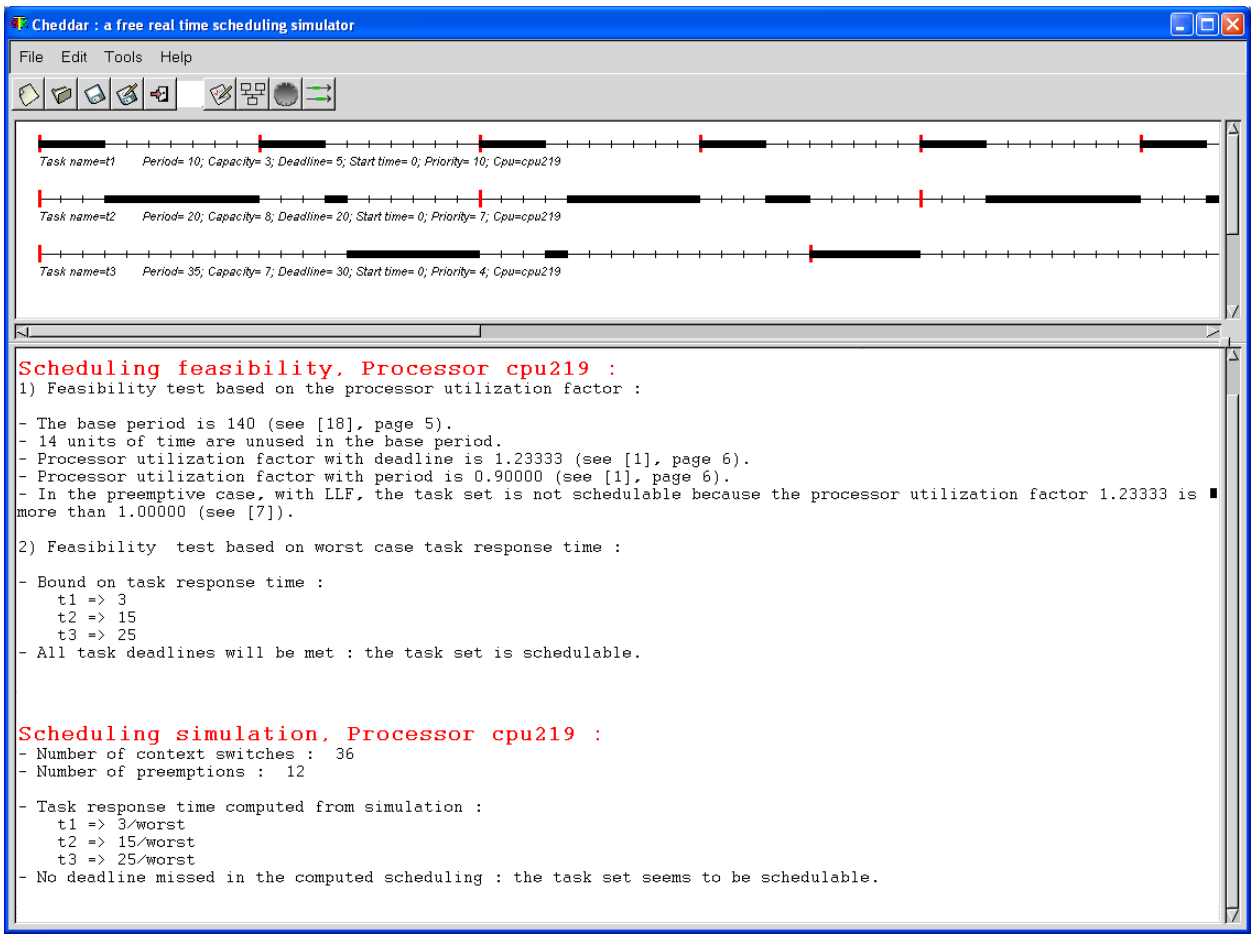
Σχήμα 3.36 Προσομοίωση σχήματος 3.11 αλγόριθμος EDF.



Σχήμα 3.37 Προσομοίωση σχήματος 3.11 αλγόριθμος RM.



Σχήμα 3.38 Προσομοίωση σχήματος 3.19 αλγόριθμος LLF.



Σχήμα 3.39 Προσομοίωση σχήματος 3.19 αλγόριθμος LLF

4. Περιγραφή των κυριότερων λειτουργικών συστημάτων πραγματικού χρόνου.

Σε αυτό το κεφάλαιο θα περιγράψουμε τα χαρακτηριστικά των κυριότερων λειτουργικών συστημάτων πραγματικού χρόνου. Στην πρώτη παράγραφο έχουμε συμπεριλάβει τα πρότυπα POSIX, OSE και μTRON. Στη δεύτερη παράγραφο το εμπορικό λογισμικό OSE, OSEK/VDX και QNX. Στην τρίτη παράγραφο το λειτουργικό σύστημα eCOS από το χώρο του ανοικτού λογισμικού και πως το Linux τροποποιήθηκε για να υποστηρίζει εφαρμογές πραγματικού χρόνου.

Η ιστοθέση των εταιρειών που αναπτύσσει και υποστηρίζει το κάθε προϊόν και ήταν η κύρια πηγή πληροφοριών αναγράφεται σε παρένθεση δίπλα στο RTOS που περιγράφουμε. Στο παράρτημα Ι υπάρχει λεπτομερής πίνακας με τις διευθύνσεις εταιρειών και τα προϊόντα τους.

4.1 Προδιαγραφές λειτουργικών συστημάτων πραγματικού χρόνου

Σε αυτή την παράγραφο περιγράφουμε τα πρότυπα POSIX, OSEK και TRON.

4.1.1 POSIX (http://standards.ieee.org/develop/wg/POSIX.html),

Το POSIX είναι μια οικογένεια προτύπων για υπηρεσίες λειτουργικών συστημάτων. Είναι σχεδιασμένο έτσι ώστε προγράμματα εφαρμογών, που ακολουθούν αυτό το πρότυπο, να μπορούν εύκολα να μεταφέρονται μεταξύ διαφορετικών λειτουργικών συστημάτων τα οποία επίσης ακολουθούν αυτό το πρότυπο[17].

Το όνομα POSIX(Portable Operating System Interface) έχει καθιερωθεί από την IEEE. Η ομάδα του POSIX αρχικά πρότεινε το πρότυπο 1003.1 που αφορά ένα σύνολο από βιβλιοθήκες διαδικασιών που πρέπει να παρέχει ένα οποιοδήποτε σύστημα UNIX, που ακολουθεί αυτό το πρότυπο. Επίσης ορίζει ποιές είναι οι παράμετροι γι' αυτές, τι πρέπει να κάνουν και ποιά αποτελέσματα επιστρέφουν. Δηλαδή η διασύνδεση των προγραμμάτων του χρήστη με τον πυρήνα, ορίζεται σε επίπεδο βιβλιοθήκης και όχι σε επίπεδο κλήσης συστήματος. Άλλες μετέπειτα εκδόσεις του POSIX προσδιορίζουν και υιοθετούν επεκτάσεις πραγματικού χρόνου και πολυνηματισμό (multithread), με αποτέλεσμα τα συστήματα που είναι συμβατά με το POSIX να χρησιμοποιούνται ευρέως στις εφαρμογές πραγματικού χρόνου.

Πρότυπα POSIX[18]

Το POSIX όπως αναφέρθηκε είναι μια οικογένεια προτύπων. Περιλαμβάνει πάνω από 30 ανεξάρτητα πρότυπα που ποικίλουν από προδιαγραφές για βασικές εφαρμογές λειτουργικών συστημάτων μέχρι προδιαγραφές για έλεγχο της συμμόρφωσης ενός λειτουργικού συστήματος σε ένα πρότυπο. Τα πρότυπα συνεχώς εμπλουτίζονται και επεκτείνονται. Παρακάτω περιγράφουμε σύντομα μερικά από τα πρότυπα του POSIX:

1003.1a: Βασικοί ορισμοί. Προσδιορίζει βασικές διεπαφές λειτουργικών συστημάτων, υποστηρίζει απλές διεργασίες και πολλαπλές διεργασίες, σήματα, ομάδες χρηστών, συστήματα αρχείων, συσκευές εισόδου-εξόδου.

1003.1b: Επεκτάσεις πραγματικού χρόνου. Παρέχει επεκτάσεις πραγματικού χρόνου, δηλαδή συναρτήσεις που χρειάζονται για λειτουργικά συστήματα πραγματικού χρόνου, συμπεριλαμβανομένων σημάτων πραγματικού χρόνου, ασύγχρονες λειτουργίες εισό-

δου-εξόδου, προτεραιότητες εισόδου-εξόδου, κλείδωμα μνήμης, προστασία μνήμης, διανομή μηνυμάτων, σηματοφορείς και κοινή μνήμη.

1003.1c: Συναρτήσεις νημάτων για υποστήριξη πολλαπλών νημάτων μέσα σε μία διεργασία. Περιλαμβάνει υποστήριξη για έλεγχο νημάτων, συμπεριφορά νημάτων, αμοιβαία αποκλειόμενες μεταβλητές και κληρονομιά προτεραιότητας αυτών, μεταβλητές υπό συνθήκη.

1003.1d: Πρόσθετες επεκτάσεις πραγματικού χρόνου. Παρέχει υποστήριξη για αποτελεσματική δημιουργία διεργασιών, σποραδικό χρονοδιάγραμμα εξυπηρετητή, παρακολούθηση χρόνων εκτέλεσης για διεργασίες και νήματα, έλεγχο διακοπών και έλεγχο συσκευών.

1003.1j: Προχωρημένες επεκτάσεις πραγματικού χρόνου. Παρέχει ακόμα περισσότερες συναρτήσεις συμπεριλαμβανομένων υποστήριξη για συγχρονισμό, κλείδωμα για ανάγνωση-εγγραφή.

1003.1h: Υψηλής διαθεσιμότητας υπηρεσίες για αξιόπιστα, διαθέσιμα και αποδοτικά συστήματα. Υποστηρίζει: καταχώρηση, τερματισμό, επανεκκίνηση και άλλα.

Βασικές υπηρεσίες POSIX.

Η IEEE POSIX 1003.1b (στο παρελθόν έκδοση 1003.4) παρέχει τα τυποποιημένα κριτήρια συμβατότητας για τις υπηρεσίες των λειτουργικών συστημάτων πραγματικού χρόνου και έχει ως σκοπό να επιτρέψει στους προγραμματιστές εφαρμογών να γράφουν εφαρμογές που μπορούν εύκολα να χρησιμοποιηθούν από όλα τα λειτουργικά συστήματα. Οι βασικές υπηρεσίες των λειτουργικών συστημάτων πραγματικού χρόνου που καλύπτονται από POSIX 1003.1b περιλαμβάνουν[18]:

Ασύγχρονο I/O (Asynchronous I/O) : Η δυνατότητα να επικαλυφθεί η επεξεργασία μιας εφαρμογής και διαδικασίες εισόδου-εξόδου.

Σύγχρονο I/O (Synchronous I/O) : Η δυνατότητα να εξασφαλισθεί η αναμονή της εκτέλεσης της διεργασίας όσο λαμβάνουν χώρα οι διαδικασίες εισόδου–εξόδου.

Κλείδωμα μνήμης (Memory locking) : Η ικανότητα να εγγυηθεί ελεύθερη μνήμη για όλες τις εκτελούμενες διεργασίες με την αποθήκευση των τμημάτων μιας διαδικασία στην οποία δεν έγινε πρόσφατα αναφορά σε συσκευές δευτερεύουσας μνήμης.

Σηματοφορείς (Semaphores) : Η δυνατότητα συγχρονισμού της πρόσβασης των πόρων από πολλαπλές διαδικασίες. Το POSIX παρέχει σηματοφόρους αρίθμησης και ψηφιακούς σηματοφόρους για να επιτρέψουν στις διαδικασίες που εκτελούνται σε διαφορετικούς χώρους διευθύνσεων ή διαφορετικά νήματα μέσα στον ίδιο χώρο διευθύνσεων, να συγχρονίζονται και να επικοινωνούν χρησιμοποιώντας κοινή μνήμη.

Κοινόχρηστη μνήμη (Shared memory) : Η δυνατότητα να αντιστοιχηθεί ένα κοινό φυσικό διάστημα μνήμης σε διαδικασίες, που η κάθε μία έχει το δικό της χώρο διευθύνσεων, για να μοιράζονται μεγάλο όγκο δεδομένων.

Χρονοπρογραμματισμός εκτέλεσης (Execution scheduling) : Η δυνατότητα να χρονοπρογραμματισμού πολλαπλών εργασιών. Οι κοινές μέθοδοι περιλαμβάνουν την εξυπηρέτηση εκ περιτροπής και τον χρονοπρογραμματισμό βασισμένο σε προτεραιότητες με προεκτόπιση.

Χρονομετρητές (Timers) : Βελτιώνουν τη λειτουργικότητα και τον ντετερμινισμό του συστήματος. Ένα σύστημα πρέπει να έχει τουλάχιστον μια συσκευή ρολογιού (ρολόι συστήματος) για να παρέχει σωστές υπηρεσίες πραγματικού χρόνου. Το ρολόι

συστήματος καλείται ρολόι πραγματικού χρόνου όταν υποστηρίζει το σύστημα πραγματικού χρόνου POSIX.

Ενδοδιεργασιακή επικοινωνία(Interprocess communication IPC): Η ενδοδιεργασιακή επικοινωνία είναι ένας μηχανισμός μέσω του οποίου οι εργασίες μοιράζονται τις πληροφορίες που απαιτούνται σε μια συγκεκριμένη εφαρμογή. Οι κοινές μέθοδοι επικοινωνίας περιλαμβάνουν τις γραμματοθυρίδες και τις ουρές.

Αρχεία πραγματικού χρόνου (Real-time files) : Η δυνατότητα δημιουργίας και επεξεργασίας αρχείων με ντετερμινιστικό τρόπο.

Νήματα πραγματικού χρόνου (Real-time threads) : Τα νήματα πραγματικού χρόνου είναι προγραμματιζόμενες οντότητες μιας εφαρμογής πραγματικού χρόνου. Έχουν μεμονωμένους περιορισμούς χρόνου ενώ όταν ανήκουν σε ένα εκτελέσιμο σύνολο νημάτων μπορούν να έχουν συλλογικούς περιορισμούς χρόνου .Οι προδιαγραφές του προτύπου καθορίζουν τη διεπαφή προγραμματισμού εφαρμογών για την δημιουργία, τον έλεγχο και τον συγχρονισμό των νημάτων καθώς και την λειτουργικότητα που πρέπει να προσφέρει μία βιβλιοθήκη νημάτων σε ένα συμβατό με το πρότυπο POSIX λειτουργικό σύστημα.

Στο πρότυπο IEEE/ANSI Std. 1003.13 (Draft Standard for Information Technology. Standardized Application Environment Profile. POSIX RealTime Application Support) περιγράφονται 4 διαφορετικά προφίλ διαφορετικής πολυπλοκότητας για εφαρμογές με απαιτήσεις πραγματικού χρόνου.

α. **Minimal Realtime System Profile**. Αυτό το προφίλ περιγράφει τον απλούστερο τύπο ενός συστήματος πραγματικού χρόνου. Το προγραμματιστικό μοντέλο περιλαμβάνει ένα επεξεργαστή - μία διεργασία με ένα η περισσότερα νήματα χωρίς μονάδα διαχείρισης μνήμης. Δεν απαιτείται σύστημα διαχείρισης αρχείων και οι συσκευές εισόδου και εξόδου λειτουργούν είτε με απεικόνιση στη μνήμη είτε με ειδικές διεπαφές.

β. **Realtime Controller System Profile**. Είναι επέκταση του minimal profile στο οποίο έχουν προστεθεί ένα σύστημα διαχείρισης αρχείων και υποστήριξη ασύγχρονης επικοινωνίας. Συσκευές μαζικής αποθήκευσης δεν είναι απαραίτητες γιατί, αν υπάρχει αρκετή μνήμη, το σύστημα αρχείων υλοποιείται στην κύρια μήνη.

γ. **Dedicated Realtime System Profile**. Είναι επέκταση του minimal profile στο οποίο έχουν προστεθεί υποστήριξη για εκτέλεση πολλών διεργασιών. Μία διεπαφή για αρχεία είναι απαραίτητη όχι όμως και ιεραρχικό σύστημα αρχείων. Καθώς πολλές διεργασίες μοιράζονται την μνήμη είναι απαραίτητος ένας μηχανισμός κλειδώματος. Αυτό το προφίλ υποθέτει ένα ή περισσότερους επεξεργαστές με ή χωρίς μονάδα διαχείρισης κύριας μνήμης.

δ. **Multi-Purpose Realtime System Profile**. Αυτό το προφίλ περιλαμβάνει όλες τις προηγούμενες λειτουργίες. Ο σκοπός είναι να μπορεί να εκτελεί μία μείξη από διεργασίες με και χωρίς απαιτήσεις πραγματικού χρόνου. Το υλικό θα πρέπει να περιλαμβάνει ένα ή περισσότερους επεξεργαστές, συσκευές δικτύωσης, συσκευές απεικόνισης, μονάδα αποθήκευσης κ.α.

Ανάλογα με το βαθμό που είναι συμβατά με το πρότυπο τα λειτουργικά συστήματα χαρακτηρίζονται είτε ως πλήρως συμβατά είτε ως μερικώς συμβατά. Στην ιστοθέση της IEEE μπορεί να βρει κάποιος τον κατάλογο με τα πιστοποιημένα λειτουργικά συστήματα. Το POSIX είναι ευρύτατα διαδεδομένο τα τελευταία 15 χρόνια. Ένα σημαντικό τμήμα του προτύπου αυτού έχει σκοπό την παροχή μεταφερσιμότητας (portability) στις εφαρμογές με απαιτήσεις πραγματικού χρόνου.

Τα προφίλ των περιβαλλόντων εφαρμογών έχουν προτυποποιηθεί κάτι που επιτρέπει στους προγραμματιστές να αναπτύσσουν λειτουργικά συστήματα πραγματικού χρόνου

συμβατά με POSIX που αρχίζει από μικρούς ενσωματωμένους πυρήνες και φτάνει σε λειτουργικά συστήματα πραγματικού χρόνου μεγάλης κλίμακας. Το πρότυπο αυτό καθορίζει την διεπαφή σε διαφορετικές γλώσσες προγραμματισμού. Συγκεκριμένα οι διεπαφές πραγματικού χρόνου καθορίζονται κυρίως σε γλώσσα C και Ada. Η λειτουργικότητα που καθορίζεται στο πρότυπο είναι παρόμοια με αυτή που έχουν τα περισσότερα από τα σύγχρονα εμπορικά λειτουργικά συστήματα και οι πυρήνες. Το πρότυπο POSIX επιτρέπει να υλοποιηθούν συστήματα τα οποία ικανοποιούν τις απαιτήσεις πραγματικού χρόνου και να είναι εύκολα μεταφέρσιμα σε διαφορετικές πλατφόρμες υλικού.

4.1.2 OSEK/VDX (www.osek-vdx.org[19]).

Ο OSEK ιδρύθηκε το 1993 σαν ένα κοινό έργο της γερμανικής αυτοκινητοβιομηχανίας, με στόχο ένα βιομηχανικό πρότυπο για μια ανοικτή αρχιτεκτονική για τις κατανεμημένες ηλεκτρονικές μονάδες ελέγχου (Electronic Control Units) σε οχήματα. Το OSEK είναι μια συντομογραφία του γερμανικού όρου «Offene Systeme und Deren Schnittstellen für die im Kraftfahrzeug Elektronik». Οι αρχικοί εταίροι του έργου ήταν η BMW, η Bosch, η DaimlerChrysler, η Opel, η Siemens, η VW και το Πανεπιστήμιο της Καρλσρούης ως συντονιστής. Οι PSA και Renault εντάχθηκε στον OSEK το 1994 και εισήγαγαν τη δική τους προσέγγιση που ήταν η VDX (Vehicle Distributed eXecutive). Το 1995 στην πρώτη συνάντηση εργασίας, η ομάδα OSEK/VDX παρουσίασε τα αποτελέσματα των εναρμονισμένων προδιαγραφών μεταξύ του OSEK και του VDX. Το 1997, μετά τη δεύτερη συνάντηση εργασίας εκδόθηκε η δεύτερη έκδοση των προδιαγραφών [20].

Η ανοιχτή αρχιτεκτονική που εισήγαγε το OSEK/VDX περιλαμβάνει τους τομείς:

- Επικοινωνίας (ανταλλαγή δεδομένων εντός και μεταξύ των μονάδων ελέγχου).
- Λειτουργικό Σύστημα (εκτέλεσης του λογισμικού των ηλεκτρονικών μονάδων ελέγχου σε πραγματικό χρόνο και βάση για τα άλλα τμήματα του OSEK/VDX).
- Διαχείριση Δικτύων (προσδιορισμός, διαμόρφωση και παρακολούθηση).

Για τις επικοινωνίες και το λειτουργικό σύστημα, υπάρχουν εναλλακτικά πρότυπα για τις κανονικές απαιτήσεις (γενικά λειτουργικό σύστημα – προδιαγραφές επικοινωνιών), ή ειδικές απαιτήσεις για αρχιτεκτονικές κάλυψης γενικού συγχρονισμού με αντοχή σε σφάλματα (προδιαγραφές OSEKtime). Όταν χρησιμοποιείται το OSEKtime, τα σήματα του έχουν υψηλότερη προτεραιότητα από τις διεργασίες του OSEK.

Μερικές από τις προδιαγραφές του OSEK/VDX έχουν τυποποιηθεί στο πρότυπο ISO 17356. Η ομάδα τυποποίησης AUTOSAR(AUTomotive Open System ARchitecture)[21] χρησιμοποιεί τις προδιαγραφές του OSEK.

Το κίνητρο για τη δημιουργία του προτύπου ήταν τα υψηλά έξοδα για την ανάπτυξη και τη διαχείριση του λογισμικού μίας ηλεκτρονικής μονάδας ελέγχου και η ασυμβατότητα παρόμοιων μονάδων που κατασκευάζονταν από διαφορετικούς κατασκευαστές λόγω διαφορετικών διεπαφών και πρωτοκόλλων που υποστήριζε ο κάθε κατασκευαστής. Ο στόχος ήταν η υποστήριξη της φορητότητας και της επαναχρησιμοποίησης του λογισμικού εφαρμογών με τυποποίηση των διεπαφών, την τυποποίηση των διεπαφών χρήστη ανεξάρτητα από το υλικό και το δίκτυο, τον αποτελεσματικό σχεδιασμό της αρχιτεκτονικής και τον έλεγχο, σε επιλεγμένα πιλοτικά έργα, της λειτουργικότητας της υλοποίησης των πρωτοτύπων.

Στον πίνακα 3.1 φαίνεται η κατάσταση των προδιαγραφών. Αυτή που κύρια ενδιαφέρει είναι το τμήμα OSEK OS που δίνει τις προδιαγραφές για το λειτουργικό σύστημα με απαιτήσεις πραγματικού χρόνου και περιγράφεται λεπτομερέστερα στην παράγραφο 3.2.2. [20].

Πίνακας 4.1. Τα πρότυπα OSEK.

OSEK OS	Version 2.2.3
OSEK COM	Version 3.0.3
OSEK NM	Version 2.5.3
OSEK Implementation Language (OIL)	Version 2.5
OSEK RTI (ORTI) Part A	Version 2.2
OSEK RTI (ORTI) Part B	Version 2.2
Binding Document	Version 1.4.2
OSEK/VDX time triggered operating system	Version 1.0
OSEK/VDX fault tolerant communication	Version 1.0

Οι ειδικές απαιτήσεις από ένα λειτουργικό σύστημα σύμφωνα με το OSEK, προκύπτει από το πλαίσιο εφαρμογής της ανάπτυξης λογισμικού για τις μονάδες ελέγχου της αυτοκινητοβιομηχανίας. Οι ακόλουθες απαιτήσεις διευθετούν χαρακτηριστικά όπως η αξιοπιστία, η ικανότητα λειτουργίας σε αυστηρό πραγματικό χρόνο και το κόστους:

- Το λειτουργικό σύστημα OSEK ρυθμίζεται και κλιμακώνεται στατικά. Ο χρήστης καθορίζει στατικά τον αριθμό των εργασιών, των πόρων και των υπηρεσιών που απαιτούνται.
- Οι προδιαγραφές του λειτουργικού συστήματος OSEK υποστηρίζουν εφαρμογές που μπορούν να λειτουργούν σε ROM, δηλαδή ο κώδικας μπορεί να εκτελεστεί από μνήμη μόνο ανάγνωση.
- Το λειτουργικό σύστημα OSEK υποστηρίζει φορητότητα των εργασιών μίας εφαρμογής.
- Οι προδιαγραφές του λειτουργικού συστήματος OSEK παρέχουν ένα ντετερμινιστικό περιβάλλον και μια αυστηρά τεκμηριωμένη συμπεριφορά για να μπορέσουν οι υλοποιήσεις του λειτουργικού συστήματος να πληρούν τις απαιτήσεις των αυτοκινήτων για πραγματικό χρόνο.
- Οι προδιαγραφές του λειτουργικού συστήματος OSEK επιτρέπουν ντετερμινιστικά την υλοποίηση των παραμέτρων απόδοσης.

Το λειτουργικό σύστημα OSEK παρέχει ένα σύνολο διαφορετικών υπηρεσιών και μηχανισμών διεργασιών. Είναι χτισμένο σύμφωνα με τις ρυθμίσεις του χρήστη στο στάδιο της δημιουργίας συστήματος.

Οι τέσσερις κατηγορίες συμμόρφωσης (conformance classes) που περιγράφονται έχουν ως στόχο να ικανοποιήσουν διαφορετικές απαιτήσεις σχετικά με τη λειτουργικότητα και την ικανότητα του λειτουργικού συστήματος OSEK. Έτσι, ο χρήστης μπορεί να προσαρμόσει το λειτουργικό σύστημα για ένα συγκεκριμένο έργο ελέγχου και το αντίστοιχο υλικό στο οποίο θα «τρέξει». Το λειτουργικό σύστημα δεν μπορεί να τροποποιηθεί αργότερα κατά το χρόνο εκτέλεσης.

Οι εφαρμογές που έχουν γραφτεί για μια ορισμένη κατηγορία συμμόρφωση πρέπει να είναι μεταφέρσιμες για OSEK υλοποιήσεις της ίδιας κατηγορίας. Αυτό διασφαλίζεται με τον ορισμό των υπηρεσιών, το πεδίο εφαρμογής των δυνατότητες τους και τη συμπεριφορά της κάθε κατηγορία συμμόρφωσης. Μόνο εάν όλες οι υπηρεσίες μιας κατηγορίας συμμόρφωσης που προσφέρονται με το καθορισμένο πεδίο εφαρμογής των δυνατότητες τους, τότε η υλοποίηση αυτού του λειτουργικού συστήματος είναι σύμφωνο με πρότυπο OSEK.

Οι ομάδες υπηρεσιών είναι δομημένες κατά λειτουργικότητα.

Διαχείριση των εργασιών (Task management).

- Ενεργοποίηση και τερματισμός εργασιών.
- Διαχείριση της κατάστασης των εργασιών και εναλλαγή εργασιών.

Συγχρονισμός (Synchronization).

Το λειτουργικό σύστημα υποστηρίζει δύο τρόπους συγχρονισμού των εργασιών.

- Την διαχείριση των πόρων για έλεγχο πρόσβασης στις μη διαχωρίσιμες λειτουργίες για την από κοινού χρήση πόρων ή συσκευών ή για τον έλεγχο της ροής του προγράμματος.
- Έλεγχος συμβάντων (events). Διαχείριση των συμβάντων για το συγχρονισμό των εργασιών.

Διαχείριση σημάτων διακοπής (Interrupt managment).

- Υπηρεσίες για την εξυπηρέτηση των σημάτων διακοπής.

Σήματα συναγερμού (Alarms).

- Σχετικοί και απόλυτοι συναγερμοί.

Χειρισμός επικοινωνίας μεταξύ επεξεργαστών(Intraprocessor message handling).

- Υπηρεσίες για την ανταλλαγή δεδομένων.

Χειρισμός σφάλματος (Error treatment).

- Μηχανισμοί υποστήριξης του χρήστη σε περίπτωση που συμβούν διάφορα σφάλματα.

4.1.3 µITRON (http://www.t-engine.org/ [22]).

Το TRON (The Real-time Operating system Nucleus) περιγράφει τις προδιαγραφές για τη σχεδίαση ενός ανοιχτού λειτουργικού συστήματος πραγματικού χρόνου. Το έργο ξεκίνησε το 1984, από τον καθηγητή Δρ Ken Sakamura στο Πανεπιστημίου του Τόκιο. Ο στόχος του έργου ήταν να δημιουργήσει μια ιδανική αρχιτεκτονική για υπολογιστές και δίκτυα και να παρασχεθεί για χρήση σε όλες τις ανάγκες της κοινωνίας. Το iTRON, ένα από τα προϊόντα του προγράμματος, είναι ένα από τα πλέον χρησιμοποιούμενα λειτουργικά συστήματα στον κόσμο. Χρησιμοποιείται σε δισεκατομμύρια ηλεκτρονικές συσκευές όπως κινητά τηλέφωνα, συσκευές αναπαραγωγής ήχου και στα αυτοκίνητα. Χρησιμοποιείται κυρίως από ιαπωνικές εταιρείες επειδή η τεκμηρίωσή του μέχρι πρόσφατα ήταν κύρια στα Ιαπωνικά. Τα τελευταία χρόνια το ενδιαφέρον για το iTRON αυξάνεται σε όλο τον κόσμο.

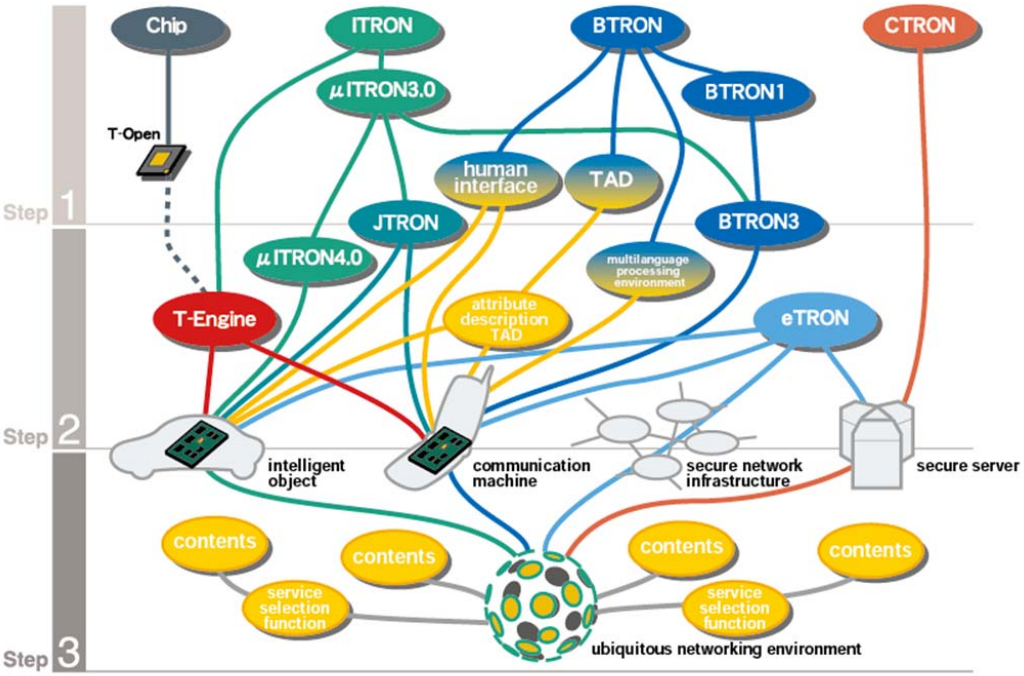
Η αρχιτεκτονική TRON είναι ένα σύνολο διασυνδέσεων και κατευθυντήριων οδηγιών σχεδίασης για τη δημιουργία ενός πυρήνα. Αυτό επιτρέπει σε διάφορες εταιρείες να δημιουργήσουν τις δικές τους εκδόσεις TRON, με βάση τις προδιαγραφές, η οποία μπορεί να είναι προσαρμοσμένη σε διαφορετικούς μικροεπεξεργαστές. Αν και οι προδιαγραφές του TRON είναι διαθέσιμες στο κοινό, οι υλοποιήσεις μπορεί να ιδιόκτητες κατά την κρίση αυτού που έκανε την υλοποίηση. Από την έναρξή του το TRON ήταν και παραμένει ανοικτή αρχιτεκτονική.

Η αρχιτεκτονική TRON ορίζει μία πλήρη αρχιτεκτονική για διάφορες μονάδες υπολογιστών:

- ITRON (Industrial TRON): μια αρχιτεκτονική για τα λειτουργικά συστήματα πραγματικού χρόνου σε ενσωματωμένα συστήματα. Αυτή είναι η πιο δημοφιλής χρή-

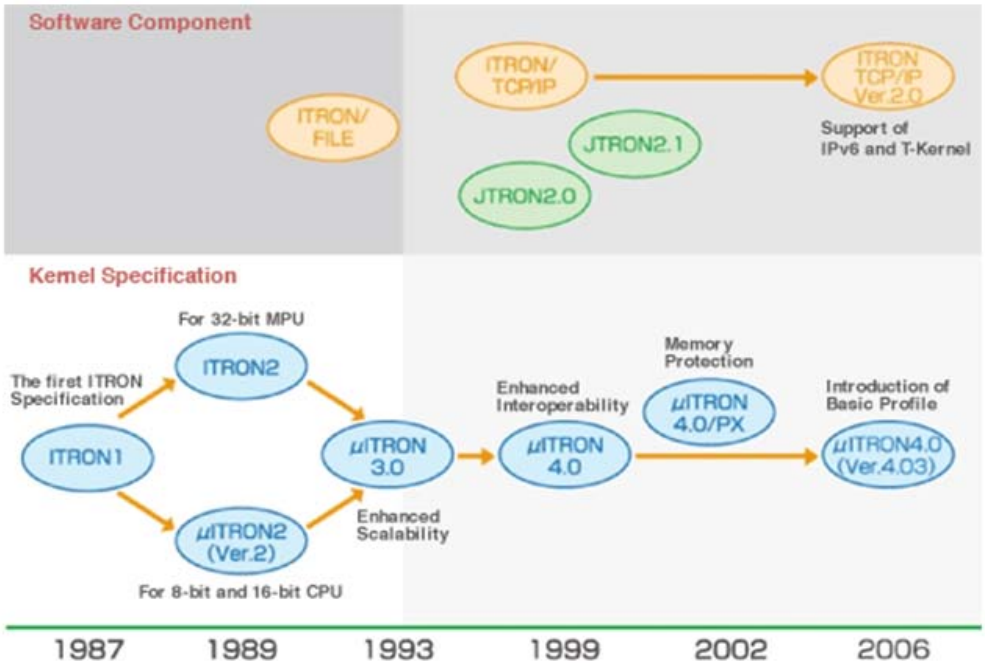
ση της αρχιτεκτονικής TRON. Το JTRON είναι ένα υποέργο της ITRON, ώστε να μπορέσει να χρησιμοποιήσει την πλατφόρμα της Java.

- BTRON (Business TRON): μια αρχιτεκτονική για τους προσωπικούς υπολογιστές, τους σταθμούς εργασίας και τα PDAs, και ως διασύνδεση ανθρώπου-μηχανής σε δίκτυα με βάση την αρχιτεκτονική TRON.
- CTRON (Central and Communications TRON): για μεγάλους υπολογιστές, και εξοπλισμό ψηφιακής μεταγωγής σε τηλεπικοινωνιακά δίκτυα.
- MTRON (Macro TRON): για την επικοινωνία μεταξύ των διαφόρων παραλλαγών τουTRON.
- STRON (Silicon TRON): Υλοποίηση σε υλικό ενός πυρήνα πραγματικού χρόνου.



Σχήμα 4.1 Το πανταχού παρόν υπολογιστικό περιβάλλον [24]

Σήμερα υπάρχουν πολλά προϊόντα, πάνω από 30, που βασίζονται σε πυρήνα πραγματικού χρόνου με βάση τις προδιαγραφές Itron για περίπου 60 επεξεργαστές. Επειδή ο πυρήνας που βασίζεται στις προδιαγραφές μITRON είναι μικρός και είναι σχετικά εύκολο να υλοποιηθεί, πολλές εταιρείες έχουν αναπτύξει τις δικές τους εκδόσεις για χρήση σε εσωτερικό επίπεδο. Υπάρχουν επίσης πολλές υλοποιήσεις πυρήνων με βάση τις προδιαγραφές μITRON που διανέμονται ως ελεύθερο λογισμικό. Ο λόγος που οι πυρήνες με βάση τις Itron-προδιαγραφές χρησιμοποιήθηκαν σε τόσες πολλές περιπτώσεις είναι επειδή υποστηρίζουν μια ευρεία γκάμα εφαρμογών και υπάρχουν πολλά πετυχημένα παραδείγματα εφαρμογής τους. Το σχήμα 4.3 δείχνει κάποιες συσκευές που χρησιμοποιούν πυρήνες προδιαγραφών Itron. Από έρευνα που πραγματοποιήθηκε πρόσφατα από την TRON Association (διενεργήθηκε από το T-Engine Forum το 2010), τα λειτουργικά συστήματα πραγματικού χρόνου που βασίζονται στις προδιαγραφές Itron χρησιμοποιούνται ευρέως σε καταναλωτικά προϊόντα και έχουν γίνει το de facto πρότυπο στη βιομηχανία. Τα γεγονός ότι πάρα πολλές επιχειρήσεις αναπτύσσουν τους δικούς τους πυρήνες προδιαγραφών Itron δηλώνει ότι οι προδιαγραφές Itron είναι πράγματι ανοικτά πρότυπα.



Σχήμα 4.2 Η εξέλιξη των προδιαγραφών μITRON [25].

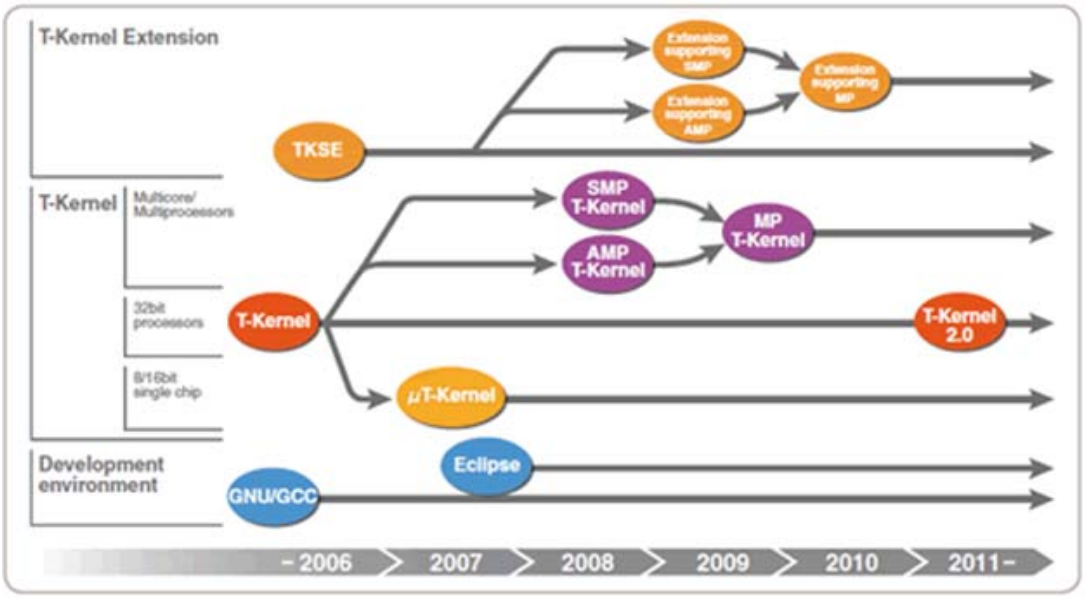
Η TRON Association που είχε την ευθύνη της διαχείρισης του έργου TRON από τις αρχές του 2010 έχει ενσωματωθεί στο T-engine Forum. Το T-Engine Forum είναι ένας μη κερδοσκοπικός οργανισμός που αναπτύσσει ανοικτές προδιαγραφές για το ITRON, τον πυρήνα T-kernel και την αρχιτεκτονική της πανταχού παρούσας ψηφιακής τεχνολογίας.

Audio/Visual Equipment, Home Appliance	TVs, Video, DVD recorders, Digital cameras, Set-top box (STBs), Audio equipment, Microwave ovens, Rice-cookers, Air conditioners, and Washing machines
Personal Information Appliance, Entertainment/Educational Equipment	PDAs, Electronic personal organizers, Car navigation systems, Game consoles, Electronic musical instruments
PC Peripheral, Office Equipment	Printers, Scanners, Disk drives, DVD drives, Copiers, FAX machines, Word processors
Communication Equipment	Answering machines, ISDN phones, Mobile phones, PHS terminals, ATM switches, Broadcasting equipment, Wireless systems, Artificial Satellites
Transportation, Industrial Control/ FA device	Automobiles, Plant Control, Industrial robots, Elevators, Vending machines, Medical equipments, and Data terminals for business

Σχήμα 4.3 Που χρησιμοποιείται το TRON [25].

Πρόεδρος του T-Engine Forum είναι ο Dr Ken Sakamura. Τον Ιούλιο του 2011 το φόρουμ είχε 266 μέλη. Η εκτελεστική επιτροπή έχει μέλη που προέρχονται από κορυφαίες Ιαπωνικές εταιρείες όπως η Fujitsu, η Hitachi, η NTT DoCoMo και η Denso. Α-επιπέδου μέλη που εμπλέκονται στο σχεδιασμό και την ανάπτυξη των προδιαγραφών της T-Engine και του T-kernel είναι εταιρείες όπως η NEC και η Yamaha Corporation. Β-επιπέδου μέλη που συμμετέχουν στην ανάπτυξη των προϊόντων που χρησιμοποιούν τις T-Engine προδιαγραφές και τον πυρήνα T-kernel είναι εταιρείες όπως η ARM, η Freescale, η MIPS Technologies, η Mitsubishi, η Robert Bosch GmbH, η Sony Corporation, η Toshiba και η Xilinx. Στο φόρουμ συμμετέχουν και πολλά πανεπιστήμια

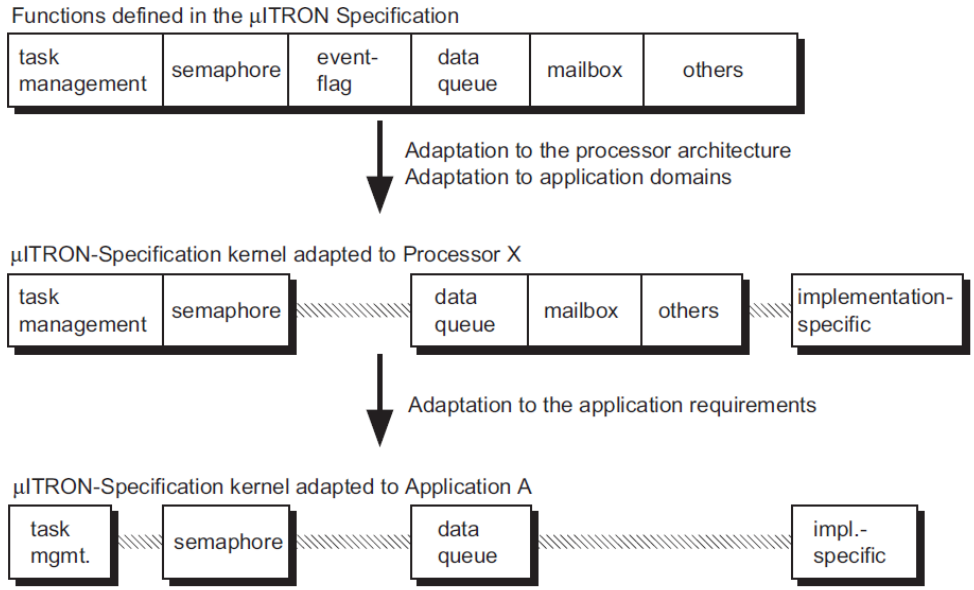
όπως το Πανεπιστήμιο του Τόκιο από την Ιαπωνία και το Dalian Maritime Πανεπιστήμιο από την Κίνα.



Σχήμα 4.4 Η εξέλιξη του T-Kernel [26].

Προδιαγραφές µTRON 4.0 [23]

Η κοινή αντίληψη των πολιτικών σχεδίασης είναι η «χαλαρή τυποποίηση». Χαλαρή τυποποίηση σημαίνει ότι κάποια τμήματα των προδιαγραφών που θα μείωναν την απόδοση του υλικού δεν είναι αυστηρά τυποποιημένα και αφήνουν ελευθερία σε αυτόν που θα κάνει την υλοποίηση για την εφαρμογή τους είτε σαν υλικό είτε σαν λογισμικό. Με την χαλαρή τυποποίηση, επιτυγχάνεται μέγιστη απόδοση για τις διάφορες πλατφόρμες υλικού, όπως φαίνεται στο σχήμα 4.5.



Σχήμα 4.5 Η υλοποίηση των προδιαγραφών µITRON.

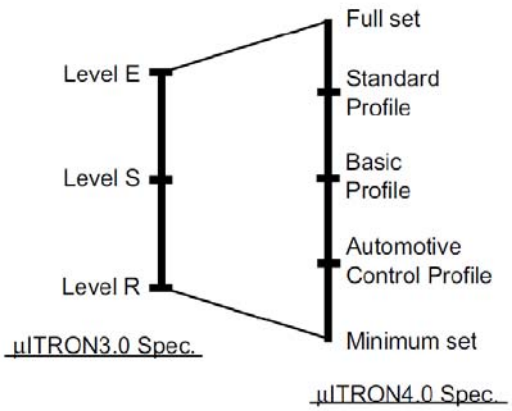
Για να βελτιωθεί η μεταφερσιμότητα του λογισμικού απαιτούνται αυστηρότερες προδιαγραφές. Οι προδιαγραφές για να βελτιώσουν την προσαρμοστικότητα στο υλικό σκόπιμα αφήνονται χαλαρές. Είναι προφανές ότι δεν υπάρχει η δυνατότητα να υποστηριχθούν ταυτόχρονα και οι δύο απαιτήσεις. Για να διευθετηθεί αυτό το θέμα έχει τυποποιηθεί ένα standard profile το οποίο αυστηρά ορίζει τις βασικές λειτουργίες και τις προδια-

γραφές τους. Ο στόχος του είναι να αυξήσει την μεταφερσιμότητα του λογισμικού διατηρώντας ταυτόχρονα τη δυνατότητα κλιμάκωσης. Το standard profile υποθέτει την ακόλουθη εικόνα:

- Επεξεργαστής 16bit ή 32bit υψηλής απόδοσης.
- Ο πυρήνας έχει μέγεθος 20Kbytes όταν περιλαμβάνει όλες τις λειτουργίες.
- Όλο το σύστημα είναι ενωμένο σε ένα τμήμα.
- Ο πυρήνας δημιουργείται σαν ένα ενιαίο αντικείμενο στατικά.

Αφού όλο το σύστημα είναι ένα ενιαίο τμήμα οι κλήσεις υπηρεσιών ενεργοποιούνται με χρήση κλήσεων υπορουτινών. Το σύστημα δεν έχει κανένα ιδιαίτερο μηχανισμό προστασίας. Οι λειτουργίες που θα υποστηρίζει περιλαμβάνουν όλες τις λειτουργίες επίπεδο S (με τροποποιήσεις και επεκτάσεις σε ορισμένες λειτουργίες) και ένα τμήμα από τις λειτουργίες του επιπέδου E (όπως με κλήσεις υπηρεσιών με χρονικό όριο, σταθερού μεγέθους περιοχές μνήμης προδιαγραφή διευθετηθεί) των προδιαγραφών μITRON3.0, αλλά και νέες λειτουργίες όπως χειρισμός εξαιρέσεων εργασιών, ουρές δεδομένων και άλλες.

Ένα προφίλ, το Automotive Control Profile, που έχει σαν στόχο τις εφαρμογές ελέγχου στην αυτοκινητοβιομηχανία έχει προστεθεί. Θεωρείται και αυτό σαν ένα σύνολο λειτουργιών που αυξάνουν τη δυνατότητα μεταφοράς του λογισμικού για συστήματα μικρότερα από αυτά του standard profile. Συγκεκριμένα λειτουργίες όπως οι λειτουργίες με χρονομετρητές, κατάσταση αναστολής, χειρισμό εξαιρέσεων μίας διεργασίας, γραμματοθυρίδες, και περιοχές μνήμης σταθερού μεγέθους είναι περιττές και έχουν παραλειφθεί. Μόνο στο Automotive Control Profile ορίζεται μια εργασία που ονομάζεται «περιορισμένο έργο». Το περιορισμένο έργο δεν μπαίνει σε κατάσταση αναμονής, έτσι περιορισμένα έργα με την ίδια προτεραιότητα, μπορούν να μοιραστούν την ίδια περιοχή στοίβας, μειώνοντας έτσι τη χρήση της μνήμης.



Σχήμα 4.6 Οι προδιαγραφές μITRON 4.0 σε σχέση με τις προδιαγραφές μITRON 3.0

Στις προδιαγραφές ορίστηκε και το Basic Profile. Το προφίλ αυτό καθορίζει τις κλήσεις εξυπηρέτησης που έχουν την ίδια λειτουργικότητα στο μITRON3.0, το μITRON4.0, και το T-kernel, με σκοπό τη διευκόλυνση της μεταφοράς των εφαρμογών μεταξύ πολλαπλών πυρήνων.

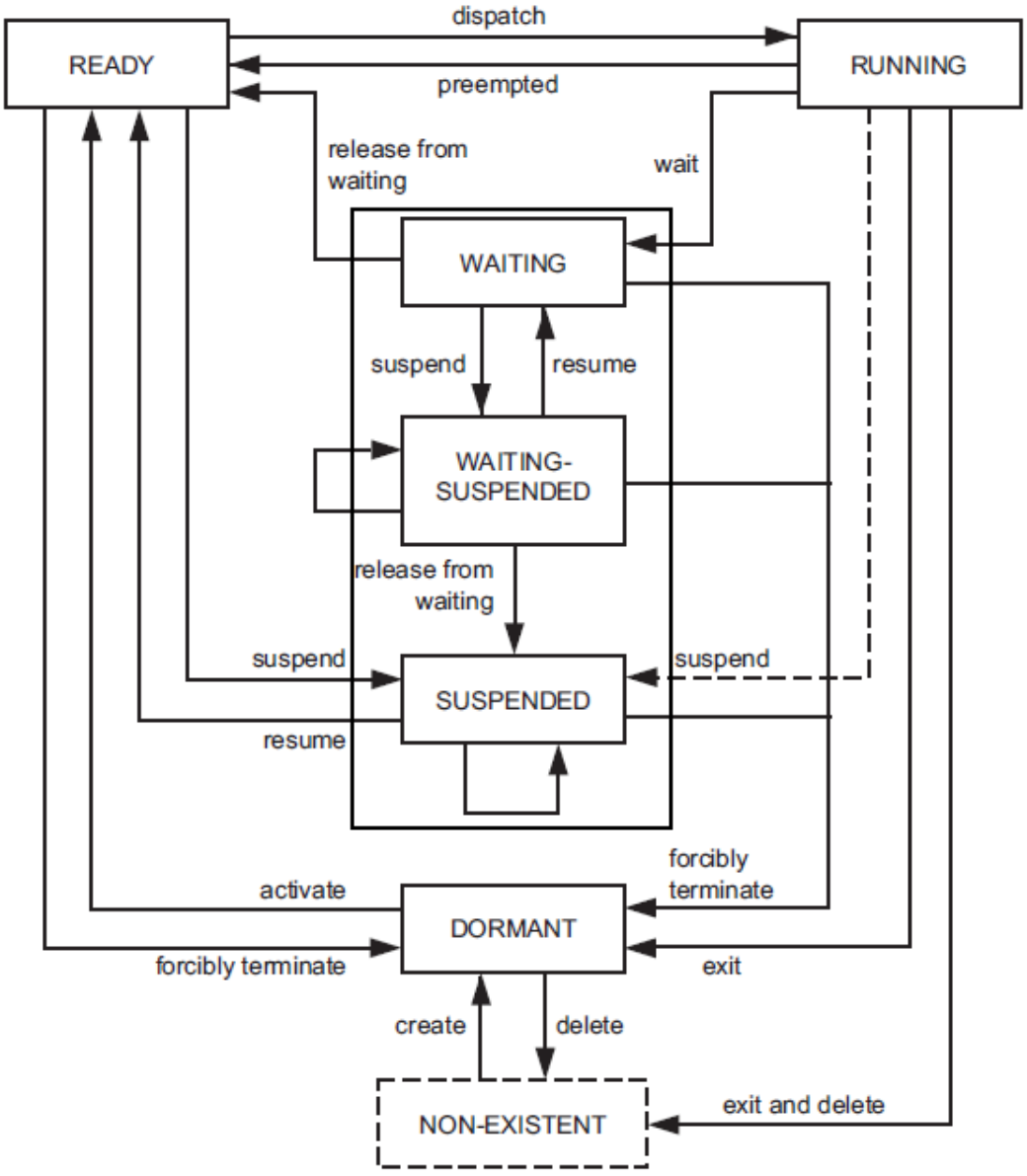
Στις προδιαγραφές 4.0 προστέθηκαν νέες λειτουργίες. Ονομαστικά αυτές οι λειτουργίες είναι οι:

- Exception Handling Functions
- Data Queues
- System State Reference Functions

- Object Creation Functions for Automatic ID Assignment
- Interrupt Service Routines
- Mutexes
- Overrun Handler
- Standard Configuration Method.

Στη συνέχεια περιγράφουμε κάποιες από αυτές. Αν κάποιος θέλει μπορεί να βρει περισσότερες λεπτομέρειες στην πηγή [23].

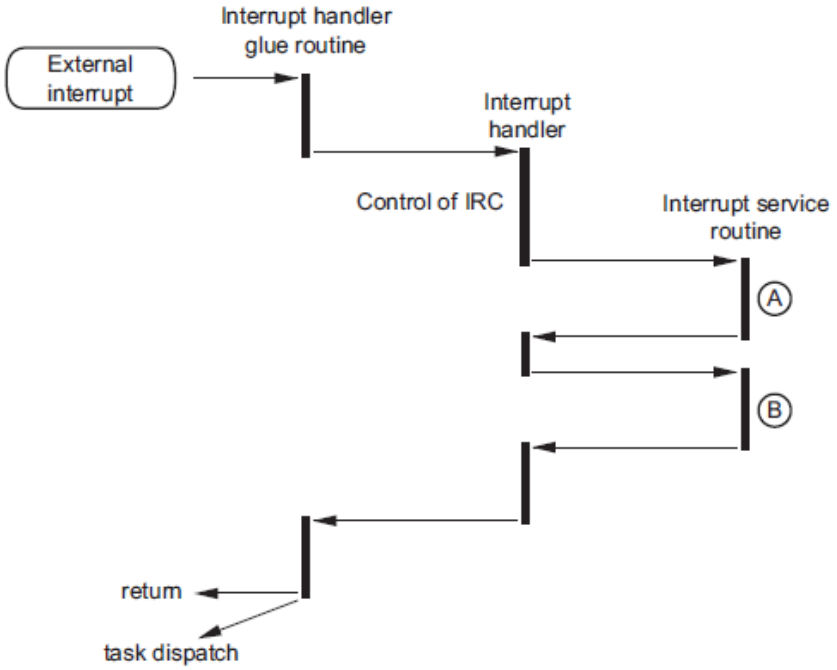
Κανόνες χρονοπρογραμματισμού και καταστάσεις διεργασίας.
Μία διεργασία μπορεί να βρίσκεται σε μία από πέντε κατηγορίες: κατάσταση RUNNING, κατάσταση READY, κατάσταση BLOCKED που έχει τρεις υποκαταστάσεις τις WAITING, SUSPENDED και WAITING-SUSPENDED, κατάσταση DORMANT και κατάσταση NON-EXISTENT. Οι καταστάσεις και οι μεταβάσεις από τη μία κατάσταση στην άλλη απεικονίζονται στο σχήμα 4.7.



Σχήμα 4.7 Διάγραμμα καταστάσεων στο μITRON 4.0

Ο χρονοπρογραμματισμός των εργασιών γίνεται με βάση τις προτεραιότητες που τους έχουν ανατεθεί. Αν υπάρχει μια σειρά από εργασίες με την ίδια προτεραιότητα, ο χρονοπρογραμματισμός γίνεται σε βάση του κανόνα First Come First Served(FCFS).

Χειρισμός διακοπών.
Στις προδιαγραφές μITRON 4.0 ο χειριστής σημάτων διακοπής και οι ρουτίνες εξυπηρέτησης σημάτων διακοπής είναι διεργασίες που αρχίζουν από κάποιο εξωτερικό σήμα διακοπής για εξυπηρέτηση. Ο χειριστής σημάτων διακοπής εξαρτάται κύρια από την αρχιτεκτονική του επεξεργαστή. Έτσι ο χειριστής σημάτων διακοπής είναι υπεύθυνος και όχι ο πυρήνας για τον έλεγχο των αιτήσεων εξυπηρέτησης. Ένας χειριστής σημάτων διακοπής δεν μπορεί να μεταφερθεί σε άλλο επεξεργαστή με διαφορετική αρχιτεκτονική. Η ρουτίνα όμως εξυπηρέτησης του σήματος διακοπής που ενεργοποιείται από τον χειριστή μπορεί να υλοποιηθεί ανεξάρτητα από την αρχιτεκτονική του επεξεργαστή.



Σχήμα 4.8 Μοντέλο χειρισμού σημάτων διακοπής.

Για να οριστεί ένα σήμα διακοπής υπάρχουν δύο τρόποι. Είτε με ένα αριθμό σήματος διακοπής είτε με ένα αριθμό ενός χειριστή διακοπών. Μία ρουτίνα εξυπηρέτησης σήματος διακοπής ορίζεται από ένα αριθμό ταυτότητα.

Χειρισμός εξαιρέσεων.
Στις προδιαγραφές μITRON 4.0 ο χειριστής εξαιρέσεων του επεξεργαστή και ο χειριστής των εξαιρέσεων των διεργασιών ορίζονται σαν λειτουργίες χειρισμού εξαιρέσεων.

Ο χειριστής εξαιρέσεων του επεξεργαστή εκκινεί όταν ανιχνεύεται μία εξαίρεση που προέρχεται από τον επεξεργαστή. Ο χειριστής εξαιρέσεων των διεργασιών χρησιμοποιείται για το σταμάτημα της κανονικής εκτέλεσης της συγκεκριμένης διεργασίας και την εκκίνηση του από κάποια κλήση εξυπηρέτησης. Η λειτουργία της ρουτίνας εξαίρεσης γίνεται στο ίδιο περιβάλλον με τη διεργασία. Μετά την ολοκλήρωση της ρουτίνας εξαίρεσης, η διακοπείσα διεργασία, μπορεί να συνεχιστεί. Ο χειριστή του συγκεκριμένου σήματος διακοπής και οι ρουτίνες εξυπηρέτησης σημάτων διακοπής είναι διεργασίες που αρχίζουν από κάποιο εξωτερικό σήμα

Λειτουργίες διαχείρισης διεργασιών.
Οι λειτουργίες διαχείρισης διεργασιών παρέχουν άμεσο έλεγχο της κατάστασης των διεργασιών και των αναφορών στην κατάσταση τους. Περιλαμβάνουν τη δυνατότητα δημιουργίας, διαγραφής μιας διεργασίας, ενεργοποίησης και τερματισμού μιας διεργασίας, την ακύρωση των αιτήσεων ενεργοποίησης, καθώς και την αναφορά της κατάστασης μίας διεργασίας.

Λειτουργίες συγχρονισμού και επικοινωνίας.
Ο συγχρονισμός και η επικοινωνία μεταξύ των διεργασιών επιτυγχάνεται με τη χρήση σηματοφορέων, σημαίες συμβάντων, ουρές δεδομένων και γραμματοθυρίδες.

Πρόσθετες λειτουργίες, που δεν είναι απαραίτητες στο standard profile, είναι οι Mutexes, οι buffer μηνυμάτων και τα randezvous ports.

4.2 Εμπορικά λειτουργικά συστήματα πραγματικού χρόνου.

Σήμερα υπάρχουν πάνω από εκατό εμπορικά προϊόντα που μπορούν να κατηγοριοποιηθούν σαν λειτουργικά συστήματα πραγματικού χρόνου, από πολύ μικρούς πυρήνες με αποτύπωμα μερικά Kilobytes μέχρι μεγάλα συστήματα για πολύπλοκες εφαρμογές πραγματικού χρόνου. Τα περισσότερα από αυτά παρέχουν υποστήριξη ταυτοχρονισμού μέσω διεργασιών ή και νημάτων. Οι διεργασίες συνήθως παρέχουν προστασία μέσω ξεχωριστών χώρων διευθύνσεων, ενώ τα νήματα μπορούν να συνεργαστούν πιο αποτελεσματικά με την κοινή χρήση του ίδιου χώρου διευθύνσεων, αλλά χωρίς προστασία. Ο χρονοπρογραμματισμός είναι συνήθως προεκτοπιστικός, διότι οδηγεί σε μικρότερες καθυστερήσεις και μεγαλύτερο βαθμό αξιοποίησης των πόρων και βασίζεται σε σταθερές προτεραιότητες. Υπάρχουν μόνο λίγα συστήματα που παρέχουν χρονοπρογραμματισμό με προτεραιότητες βασισμένες στις προθεσμίες. Οι πιο προηγμένοι πυρήνες εφαρμόζουν κάποια μορφή κληρονόμησης της προτεραιότητας (priority inheritance) για την πρόληψη της αντιστροφής προτεραιότητα (priority inversion) κατά την πρόσβαση σε αλληλοαποκλειόμενους πόρους. Αυτό απαιτεί επίσης τη χρήση ουρών προτεραιότητας, αντί της απλής ουράς FIFO.

Πολλά λειτουργικά συστήματα παρέχουν και ένα σύνολο εργαλείων για την διευκόλυνση της ανάπτυξης εφαρμογών πραγματικού χρόνου. Εκτός από τα γενικά εργαλεία προγραμματισμού, όπως συντάκτες (editors), μεταγλωττιστές (compilers) και προγράμματα εκσφαλμάτωσης (debuggers), υπάρχουν διάφορα εργαλεία που κατασκευάζεται ειδικά για την ανάπτυξη συστημάτων πραγματικού χρόνου. Προηγμένα εργαλεία περιλαμβάνουν αναλυτές μνήμη, profilers απόδοσης, παρακολούθηση σε πραγματικό χρόνο (για να βλέπουμε τις μεταβλητές, ενώ το πρόγραμμα εκτελείται), και ιχνηθέτες (tracers) εκτέλεσης (για να παρακολουθεί και να εμφανίζει τα γεγονότα του πυρήνα σε γραφική μορφή). Ένα άλλο χρήσιμο εργαλείο για την σε πραγματικό χρόνο συστημάτων είναι ο αναλυτής χρονοπρογραμματισμού, που επιτρέπει στους σχεδιαστές την επαλήθευση της εφικτότητας του χρονοπρογραμματισμού βάση διαφόρων σεναρίων σχεδιασμού. Υπάρχουν επίσης αναλυτές κώδικα για τον καθορισμό του χρόνου εκτέλεσης στην χειρότερη περίπτωση (worst-case execution times) ενός έργου σε συγκεκριμένες αρχιτεκτονικές.

Στη συνέχεια θα περιγράψουμε μερικά από τα πιο δημοφιλή εμπορικά RTOS που είναι τα OSE, OSEK/VDX και το QNX.

4.2.1 OSE – Operating System Embedded (www.enea.com).

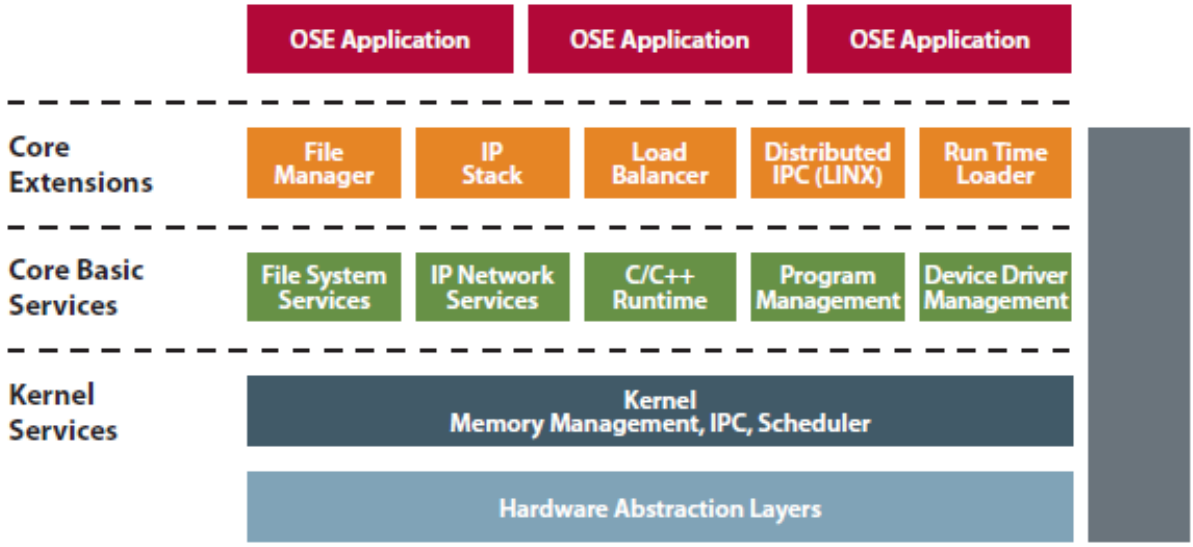
Το OSE (Operating System Embedded) εξελίσσεται από τα μέσα της δεκαετίας του 80. Αρχικός του σκοπός ήταν η δημιουργία ενός λειτουργικού συστήματος για χρήση σε

υπολογιστές ελέγχου πτήσεων αεροσκαφών. Σήμερα είναι ένα από τα πιο διαδεδομένα RTOS με χρήση κύρια στην αυτοκινητοβιομηχανία, στην ιατρική τεχνολογία και τις τηλεπικοινωνίες. Έχει μεγάλο ποσοστό της αγοράς στα κινητά τηλέφωνα και τους σταθμούς βάσης δικτύων κινητών επικοινωνιών, κύρια 3ης γενιάς. Το μεγαλύτερο μέρος του είναι γραμμένο σε γλώσσα C, ενώ κάποια τμήματά του είναι γραμμένα σε assembly και C++. Η ENEA αναπτύσσει δύο οικογένειες, το OSE και το OSEck (compact kernel) που έχει σαν στόχο την αγορά των DSPs.

Enea OSE.

Το OSE είναι βελτιστοποιημένο για χρήση σε κατανεμημένα συστήματα με υψηλή αντοχή σε σφάλματα. Είναι συμπαγές, σταθερό και υποστηρίζει μία ευρεία γκάμα εφαρμογών. Η αρχιτεκτονική του σχεδίαση σε στρώματα συμβάλλει στην υψηλή του απόδοση και την δυνατότητα κλιμάκωσης από ένα επεξεργαστή σε πολλούς επεξεργαστές και τέλος σε μεγάλα κατανεμημένα συστήματα με πολλούς πολυπύρηνους επεξεργαστές.

Δημιουργήθηκε από την αρχή με ευκολίες στην προστασίας της μνήμης, που αποτρέπουν τις εφαρμογές που δυσλειτουργούν να οδηγήσουν τον πυρήνα και τις άλλες εφαρμογές σε κατάρρευση. Ο ενσωματωμένος έλεγχος διεργασιών και η ανίχνευση σφαλμάτων απλοποιούν την ανάπτυξη και τη διόρθωση μίας εφαρμογής και καθιστούν τα κατανεμημένα συστήματα ευκολότερα στις δοκιμές, την αναβάθμιση, και την πιστοποίηση.



Σχήμα 4.9 Αρχιτεκτονική μικροπυρήνα OSE [27].

Η αρχιτεκτονική μικροπυρήνα έχει πολλά πλεονεκτήματα:

- Είναι μικρός και αποδοτικός σε πολυπύρηνα συστήματα πραγματικού χρόνου με αντοχή σε σφάλματα και προβλέψιμη συμπεριφορά.
- Έχει προηγμένα χαρακτηριστικά γνωρίσματα δικτύωσης και ασφάλειας.
- Είναι βελτιστοποιημένος για υποστήριξη σύνθετων κατανεμημένων, ομογενών και ετερογενών, πρωτοκόλλων δικτύωσης
- Παρέχει υποστήριξη σελιδοποίησης μετά από απαίτηση για τη βελτιστοποίηση της χρήσης της μνήμης RAM.
- Διαχείριση ενέργειας με πολύ χαμηλή κατανάλωση σε κατάσταση ύπνου.
- Δυναμική φόρτωση προγράμματος στο χρόνου εκτέλεσης.
- Πολλές επιλογές για σύστημα διαχείρισης αρχείων.

Η ελάχιστη τυπική διαμόρφωση του OSE απαιτεί 350Kbytes που περιλαμβάνει το στρώμα Kernel Services και το στρώμα Core Basic Services. Αν κάποιος χρήστης θέλει ακόμη μικρότερο αποτύπωμα τότε δεν έχει παρά να αφαιρέσει τα τμήματα του Core Basic Services που δεν χρειάζεται η εφαρμογή του.

Για να βελτιώσει την απόδοση το OSE χρησιμοποιεί την άμεση επικοινωνία μεταξύ των διεργασιών έτσι δεν απαιτείται συγχρονισμός μέσω ενδιάμεσων μηχανισμών. Για ευκολία και αύξηση της απόδοσης όταν τα μηνύματα που ανταλλάσσονται μεταξύ των διεργασιών μοιράζονται την ίδια μνήμη, ο πυρήνας διαβιβάζει τους δείκτες στα μηνύματα για να μην απαιτείται η αντιγραφή των μηνυμάτων που έχουν πολύ μεγαλύτερο μέγεθος.

Ο πυρήνας είναι πλήρως προεκτοπιστικός και μπορεί να παρέχει εξυπηρέτηση σε αίτημα διακοπής ακόμη και κατά την εκτέλεση μίας κλήσης συστήματος.

Το Enea LINX επεκτείνει τις υπηρεσίες διαβίβασης μηνυμάτων και την διαχείριση διεργασιών σε πολλούς επεξεργαστές και λειτουργικά συστήματα παρέχοντας την πλήρη υποστήριξη σε κατανεμημένα συστήματα. Η Enea ισχυρίζεται ότι είναι το μοναδικό σύστημα διεργασιακής επικοινωνίας που μπορεί να υποστηρίξει οποιαδήποτε τοπολογία, από ένα επεξεργαστή μέχρι δίκτυα με πολλούς επεξεργαστές σε συστοιχίες.

Το σύστημα διαχείρισης αρχείων για φορητές συσκευές είναι περίπου 200Kbytes. παρέχει όμως και υποστήριξη στο πρότυπο POSIX 1003.1 ώστε να είναι εύκολο να δει εφαρμογές UNIX. Λόγω της τμηματικής δομής του OSE ο διαχειριστής αρχείων επιτρέπει το δυναμικό φόρτωμα ή ξεφόρτωμα ενός νέου συστήματος αρχείων.

Το Enea Optima είναι μία σουίτα εργαλείων, για Windows Linux ή Solaris, στο Eclipse περιβάλλον για το OSE. Το Enea Soft Kernel Environment παρέχει ένα περιβάλλον προσομοίωσης που επιτρέπει την εκτέλεση του πυρήνα σε συστήματα Windows, Linux ή Solaris, ώστε οι προγραμματιστές να αρχίσουν την ανάπτυξη της εφαρμογής πριν το υλικό να είναι έτοιμο. Είναι ένα API που υποστηρίζει όλες τις κλήσεις συστήματος και τα πιο πολλά τμήματα του OSE.

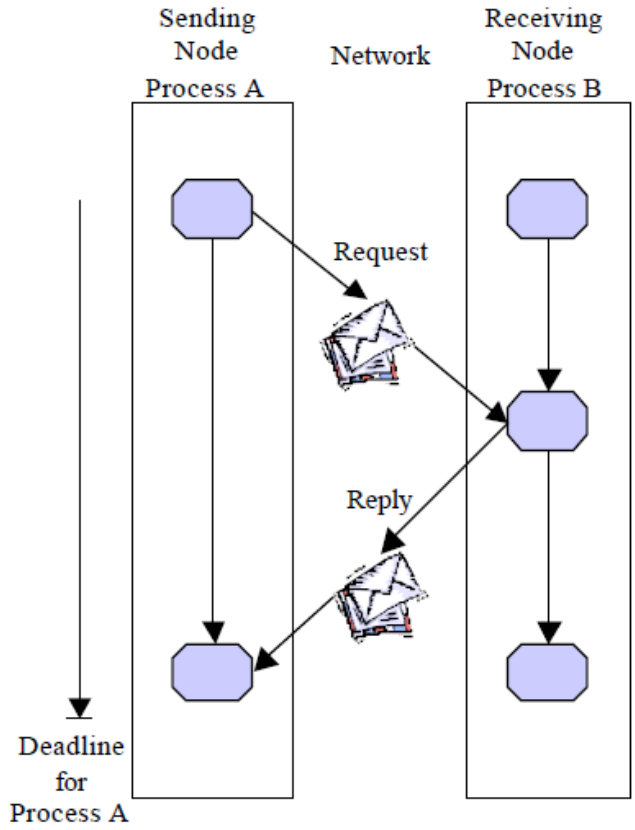
Σχεδιαστικοί στόχοι του OSE από την αρχή ήταν η αξιοπιστία, η απλότητα και η κλιμάκωση. Είναι ένα λειτουργικό σύστημα βελτιστοποιημένο για τη μεταβίβαση μηνύματος. Η μεταβίβαση μηνύματος είναι ιδιαίτερα χρήσιμη στα κατανεμημένα συστήματα όπου η επικοινωνία μεταξύ πολλών και διαφορετικών επεξεργαστών είναι αναγκαία. Ο βασικός μηχανισμός ενδοδιεργασιακής επικοινωνίας στο OSE είναι η ασύγχρονη άμεση μεταβίβαση μηνύματος. Ασύγχρονη επικοινωνία σημαίνει ότι χρησιμοποιεί κλήσεις αποστολής λήψης χωρίς μπλοκάρισμα. Άμεση μεταβίβαση σημαίνει ότι η επικοινωνία μεταξύ δύο διεργασιών γίνεται άμεσα χωρίς την μεσολάβηση κάποιας γραμματοθυρίδας. [29]

Η επικοινωνία στο OSE επιτυγχάνεται με τη χρήση σημάτων, σηματοφορέων, γρήγορων σηματοφορέων και mutexes. Τα σήματα είναι η πιο απλή μέθοδος. Μπορεί να περιέχουν και δεδομένα και ορίζονται εύκολα με τη δημιουργία μιας δομής ή μιας κλάσης. Είναι πολύ αποτελεσματικά και εργάζονται μεταξύ δύο διεργασιών σε ένα επεξεργαστή ή σε διεργασίες διαφορετικών επεξεργαστών. Τα σήματα δημιουργούν μία ουρά FIFO και το OSE μπορεί να επιλέξει ένα σήμα από οποιοδήποτε σημείο της ουράς.

Το OSE στις περισσότερες εφαρμογές χρησιμοποιεί οκτώ κλήσεις, οι δύο είναι η send και η receive. Η δόμησή του σε μικροπυρήνα επιτρέπει την υλοποίηση υπηρεσιών σαν ανεξάρτητα πρόσθετα τμήματα και όχι μέσα στον πυρήνα. Το βασικό δομικό στοιχείο του OSE είναι η διεργασία. Οι έννοιες διεργασία, νήμα και task που συναντάμε σε άλλα λειτουργικά συστήματα δεν υφίστανται στο OSE.

Ο χρονοπρογραμματιστής υλοποιεί προεκτοπιστικό χρονοπρογραμματισμό βασισμένο σε προτεραιότητες που συνδυάζεται με κυκλικό και περιοδικό χρονοπρογραμματισμό για διάφορα είδη διεργασιών. Το OSE για να βελτιώσει την δομή του συστήματος επι-

τρέπει την ομαδοποίηση σε μπλοκ των διεργασιών. Κάθε μπλοκ μπορεί να έχει τη δική του περιοχή μνήμης και αν συμβεί κάτι δεν καταρρέει ολόκληρο το σύστημα. Η μονάδα διαχείρισης της μνήμης μπορεί να απομονώνει τα διάφορα τμήματα. Αυτή η δομή φαίνεται στο σχήμα 4.12. Υπάρχει ένας τομέας συστήματος που περιέχει τον πυρήνα και μερικοί τομείς εφαρμογών που περιλαμβάνουν μία περιοχή μνήμης για την εφαρμογή και ένα σωρό για την δυναμική κατανομή μνήμης. Υπάρχει πάντα μία περιοχή μνήμης που γίνεται global allocation.



Σχήμα 4.10 Επικοινωνία με διαβίβαση μηνύματος.

Ένα μοναδικό χαρακτηριστικό του OSE είναι η διαχείριση σφάλματος. Υπάρχουν τέσσερα προκαθορισμένα επίπεδα χειρισμού σφάλματος. Το επίπεδο διεργασίας, το επίπεδο μπλοκ, το επίπεδο συστήματος και το επίπεδο πυρήνα. Όταν ένα σφάλμα ανιχνεύεται σε κάποια από τα παραπάνω επίπεδα ο αντίστοιχος χειριστής σφάλματος ενεργοποιείται. Σκοπός του χειριστή είναι είτε να διορθώσει το σφάλμα είτε να επιβεβαιώσει στον καλούντα την ύπαρξη σφάλματος. Μία άλλη εναλλακτική εργασία είναι να τερματίσει αν είναι χειριστής σφάλματος διεργασίας τη διεργασία ή το μπλοκ αν είναι χειριστής σφάλματος μπλοκ κοκ. Εάν ο χειριστής σφάλματος που κλήθηκε δεν μπορεί να επιλύσει την κατάσταση καλείται ο χειριστής σφάλματος του επόμενου επιπέδου για να προσπαθήσει να επιλύσει το πρόβλημα. Αυτή η λειτουργία και η αρχιτεκτονική της χαλαρής σύνδεσης περιοχών μνήμης κατατάσσει το OSE στα συστήματα με αντοχή στα σφάλματα.

Διαχείριση διεργασιών [30]

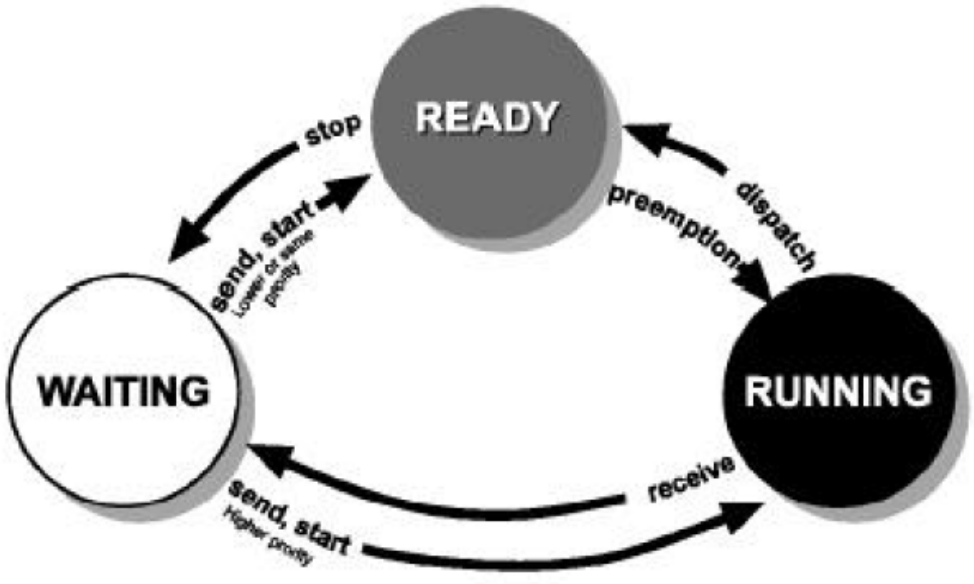
Κάθε διεργασία έχει ένα όνομα και ένα αναγνωριστικό αριθμό. Το όνομα μπορεί να χρησιμοποιηθεί από πολλές ανεξάρτητες διεργασίες, το pid όμως είναι μοναδικό και αποδίδεται από το σύστημα σε κάθε εκτελούμενη διεργασία. Αναλόγως με το πότε δημιουργήθηκε μία διεργασία είναι στατική ή δυναμική. Οι στατικές διεργασίες δηλώνονται στο

αρχείο διαμόρφωσης και αρχίζουν με την εκκίνηση του συστήματος. Αυτές οι διεργασίες δεν μπορούν να «πεθάνουν»ή να «σκοτωθούν»όσο τρέχει το σύστημα. Οι δυναμικές διεργασίες δημιουργούνται οποτεδήποτε. Μπορούν να «πεθάνουν», να «σκοτωθούν»και να αναδημιουργηθούν αργότερα.

Το OSE υποστηρίζει πέντε τύπους διεργασιών που μπορεί να χρησιμοποιήσει ο προγραμματιστής:

- Interrupt είναι μία διεργασία που τρέχει όταν ένα σήμα διακοπής ενεργοποιηθεί είτε από το υλικό είτε από το λογισμικό. Είναι οι υψηλότερης προτεραιότητας διεργασίες.
- Timer-interrupt είναι διεργασίες που τρέχουν σε περιοδικά χρονικά διαστήματα.
- Prioritized η πιο κοινή διεργασία στο OSE. Τρέχει όσο δεν είναι έτοιμη άλλη διεργασία υψηλότερης προτεραιότητας ή κάποιο σήμα διακοπής ή δεν παραιτείται από μόνη της. Μεταξύ διεργασιών ίσης προτεραιότητας εφαρμόζεται το σχήμα round robin.
- Background εκτελούνται κυκλικά με χρονομερισμό (Timesharing) κάτω από τις διεργασίες με προτεραιότητες όσο δεν υπάρχουν άλλες διεργασίες με προτεραιότητα έτοιμες.
- Phantom δεν είναι πραγματικές διεργασίες. Χρησιμοποιούνται σαν αντιπρόσωποι κάποιας διεργασίας σε ένα πίνακα ανακατεύθυνσης που ανακατευθύνει ένα εισερχόμενο σήμα σε μία ή περισσότερες διεργασίες.

Υπάρχουν 32 επίπεδα, από 0..31, με το επίπεδο 0 να αντιστοιχεί στην υψηλότερη προτεραιότητα. Υπάρχει η δυνατότητα σε κάθε διεργασία να αποδοθεί μία συγκεκριμένη προτεραιότητα. Οι διεργασίες Interrupt και timer-interrupt πρέπει να εκτελούν ένα μικρό τμήμα κώδικα, πχ την αποστολή ενός σήματος, για νε ελαχιστοποιείται ο χρόνος που τα σήματα διακοπής είναι απενεργοποιημένα.



Σχήμα 4.11 Καταστάσεις μιας διεργασίας.

Κάθε χρονική στιγμή εκτελείται η έτοιμη διεργασία με την υψηλότερη προτεραιότητα. Μία διεργασία μπορεί να βρίσκεται σε μία από τις καταστάσεις του σχήματος 4.11.

Running σε αυτή έχει αποδοθεί η cpu. Σε ένα μονοεπεξεργαστικό σύστημα μία διεργασία βρίσκεται σ' αυτή την κατάσταση.

Ready όλες οι έτοιμες διεργασίες για εκτέλεση τοποθετούνται σε μία ουρά. Σε κάθε αλλαγή διεργασίας η έτοιμη με την υψηλότερη προτεραιότητα παίρνει την cpu. Κάθε επίπεδο έχει τη δική του ουρά.

Waiting η διεργασία είτε περιμένει να συμβεί κάτι είτε έχει σταματήσει. Οι διεργασίες σε αναμονή δεν απαιτούν Cpu.

Χρονοπρογραμματισμός.
Οι αρχές χρονοπρογραμματισμού που ακολουθεί το OSE είναι¨

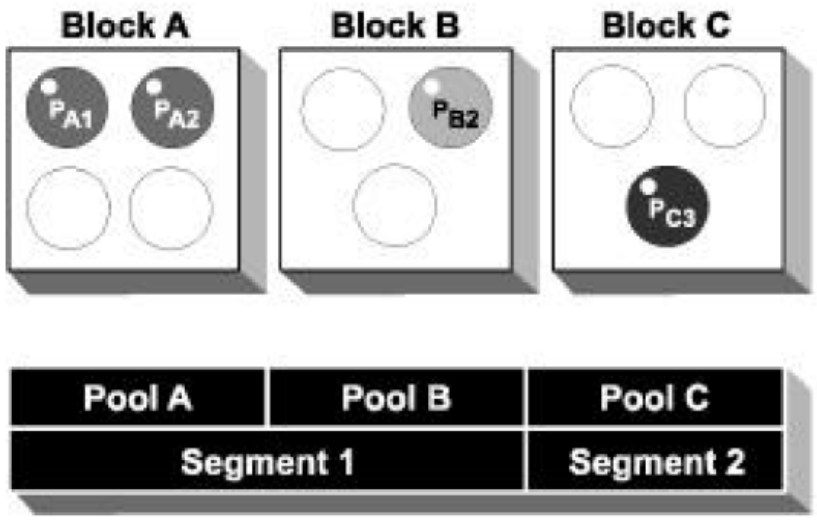
- Η προεκτόπιση. Όλες οι διεργασίες προεκτοπίζονται ακόμη και όταν εκτελούν μία κλήση συστήματος αν μία υψηλότερης προτεραιότητας είναι έτοιμη προς εκτέλεση.
- Η κυκλική. Οι timer-interupt διεργασίες χρονοπρογραμματίζονται για εκτέλεση σε συγκεκριμένα χρονικά διαστήματα.
- Η προτεραιότητα. Εκτελείται η έτοιμη διεργασία με την υψηλότερη προτεραιότητα. Αυτό ισχύει για τις διεργασίες με προτεραιότητα.

Ενδοδιεργασιακή επικοινωνία.
Ο τρόπος περάσματος δεδομένων μεταξύ διεργασιών στο OSE είναι τα σήματα. Στο OSE ένα σήμα είναι ένα μήνυμα που στέλνεται απ' ευθείας από μία διεργασία σε μία άλλη. Το μήνυμα έχει ένα αναγνωριστικό, τον αποστολέα, τον παραλήπτη και τα δεδομένα. Όταν ένα σήμα αποσταλεί, η διεργασία που το έστειλε δεν μπορεί να το προσπελάσει. Η διεργασία παραλήπτης μπορεί να ορίσει τον τύπο του σήματος που θέλει να λάβει κάποια χρονική στιγμή. Η διεργασία μπορεί είτε να περιμένει για το ειδικό σήμα είτε να περιοδεύει την ουρά των σημάτων. Την ιδιοκτησία των σημάτων την διαχειρίζεται ο πυρήνας. Η γνώση του ποιος είναι ιδιοκτήτης ενός σήματος διευκολύνει την εκσφαλμάτωση μίας εφαρμογής.

Όταν αποστέλλεται ένα σήμα η διεργασία αποστολέας μπορεί να στείλει μόνο τον δείκτη στον Buffer που περιέχει τα δεδομένα. Αν η επικοινωνία γίνεται μεταξύ διεργασιών που ανήκουν σε διαφορετικό τμήμα μπορεί να επιλέξει την αποστολή του δείκτη αλλά και την αντιγραφή των δεδομένων του μηνύματος. Τα σήματα «πεθαίνουν»μαζί με τη διεργασία ιδιοκτήτη και το λειτουργικό σύστημα αποδεσμεύει τη μνήμη.

Διαχείριση μνήμης.

Σε κάθε σύστημα πραγματικού χρόνου η διαχείριση της μνήμης είναι πολύ σημαντική. Στο OSE η μνήμη όπως φαίνεται και στο σχήμα 4.12 είναι οργανωμένη σε περιοχές μνήμης(pools) και τμήματα. Υπάρχει η περιοχή του συστήματος που είναι στο τμήμα στη μνήμη του πυρήνα. Όλες οι διεργασίες μπορούν να πάρουν μνήμη από την περιοχή του συστήματος. Το OSE δίνει περιοχές μνήμης είτε σε τοπικό επίπεδο είτε σε μπλοκ. Για να έχει το σύστημα ανοχή σε σφάλματα οι περιοχές είναι απομονωμένες ώστε αν κάποιο τμήμα καταρρεύσει να μην καταρρεύσει όλο το σύστημα. Η απόδοση των περιοχών της μνήμης γίνεται με κομμάτια προκαθορισμένου μεγέθους. Αυτό συμβάλλει στον ντετερμινισμό του συστήματος γιατί είναι εύκολη η τήρηση πληροφοριών ποια και πόσα τμήματα μνήμης είναι ελεύθερα.

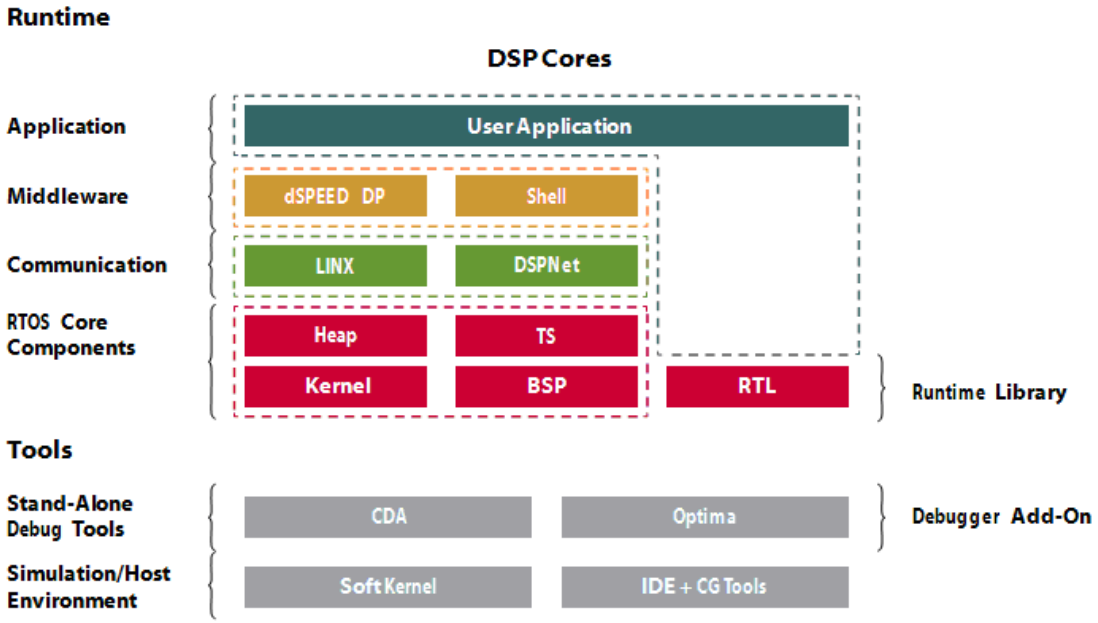


Σχήμα 4.12 Η κύρια μνήμη στο OSE.

Enea OSEck.

Το OSEck είναι μία βελτιστοποιημένη έκδοση του OSE για DSPs, κατάλληλο για εφαρμογές με περιορισμένη χρήση μνήμης.

Η οδηγούμενη από γεγονότα προεκτοπιστική λειτουργία του OSEck είναι κατάλληλη για τις εφαρμογές επεξεργασίας και ελέγχου σήματος που έχουν χρονικούς περιορισμούς. Η αρχιτεκτονική του σχεδίαση σε στρώματα και η διαβίβαση μηνύματος βοηθούν στην τμηματική σχεδίαση εφαρμογών και την μείωση της πολυπλοκότητα τους. Λόγω της δυνατότητας προσομοίωσης σε κάθε επίπεδο η ανάπτυξη του κώδικα μπορεί να γίνει πριν από το υλικό στο οποίο θα τρέξει.



Σχήμα 4.13 Η πλατφόρμα OSEck [31].

Το OSEck υποστηρίζει ένα υποσύνολο του API του OSE, καθιστώντας έτσι εύκολη τη μετανάστευση των εφαρμογών μεταξύ OSE και OSEck, με ελάχιστες αλλαγές στον κώ-

δικα της εφαρμογής. Τα DSPNet παρέχει χαρακτηριστικά δικτύωσης και ασφάλειας που εξασφαλίζουν τη δικτύωση μέσω της στοίβας πρωτοκόλλων IPv4/IPv6, IPSEC και SSH.

Το OSEck είναι ιδανικό για τις εφαρμογές με αυστηρούς περιορισμούς μνήμης που απαιτούν, τον αξιόπιστο έλεγχο σε πραγματικό χρόνο και την σε πραγματικό χρόνο επεξεργασία σήματος. Το OSEck :

Είναι σχεδιασμένο για τα κατανεμημένα ετερογενή περιβάλλοντα.

- Είναι εύκολα μεταφέρσιμο.
- Είναι βελτιστοποιημένο για μονοπύρηνους και πολυπύρηνους DSPs.
- Παρέχει εύκολη κλιμάκωση στο επίπεδο λειτουργίας, επιτρέποντας η λειτουργικότητα και το ίχνος να μπορούν να βελτιστοποιηθούν για κάθε εφαρμογής.

Πλατφόρμες υλικού
Το OSE υποστηρίζει τους επεξεργαστές

- της οικογένειας ARM (ARM 9, ARM11, ARM Cortex).
- της οικογένειας MIPS 64 (n32 ABI Netlogic XLR and XLS processors. Cavium Networks Octeon and Octeon Plus).
- της οικογένειας Freescale PowerPC (MPC 5xx, MPC 5xxx, Host Processors MPC 7xxx, PowerQUICC I MPC 8xx, PowerQUICC II MPC 82xx and MPC 83xx, PowerQUICC III MPC 85xx, Host Processors MPC 8xxx. QorIQ: P10xx, P20xx and P40xx).
- της οικογένειας IBM PowerPC –από την AMCC – (405EP, 405EXr, 405GP, 440GP, 440GRx, 440GX)
- της οικογένειας IBM PowerPC family – από την IBM- (750, 750CX, 750FX, 750GP, 750GX)
- της οικογένειας INTEL ARM 5 family (XScale)
- της οικογένειας Intel Network Processor (IXP465, IXP2350)
- της οικογένειας Texas Instruments (OMAP (all chipsets, for OMAP DSP)

4.2.2 OSEK/VDX

Αρχιτεκτονική του λειτουργικού συστήματος OSEK [20].
Το λειτουργικό σύστημα OSEK χρησιμεύει ως βάση για την εκτέλεση προγραμμάτων εφαρμογών τα οποία είναι ανεξάρτητα μεταξύ τους, και παρέχει το περιβάλλον εκτέλεσης τους σε έναν επεξεργαστή. Το λειτουργικό σύστημα OSEK επιτρέπει την σε ένα ελεγχόμενο, πραγματικό χρόνο, εκτέλεση των διαφόρων διαδικασιών που φαίνεται να εξελίσσονται παράλληλα.

Το λειτουργικό σύστημα OSEK παρέχει ένα καθορισμένο σύνολο διεπαφών για το χρήστη. Αυτές οι διασυνδέσεις χρησιμοποιούνται από οντότητες που ανταγωνίζονται για τη χρήση του επεξεργαστή. Υπάρχουν δύο τύποι οντοτήτων:

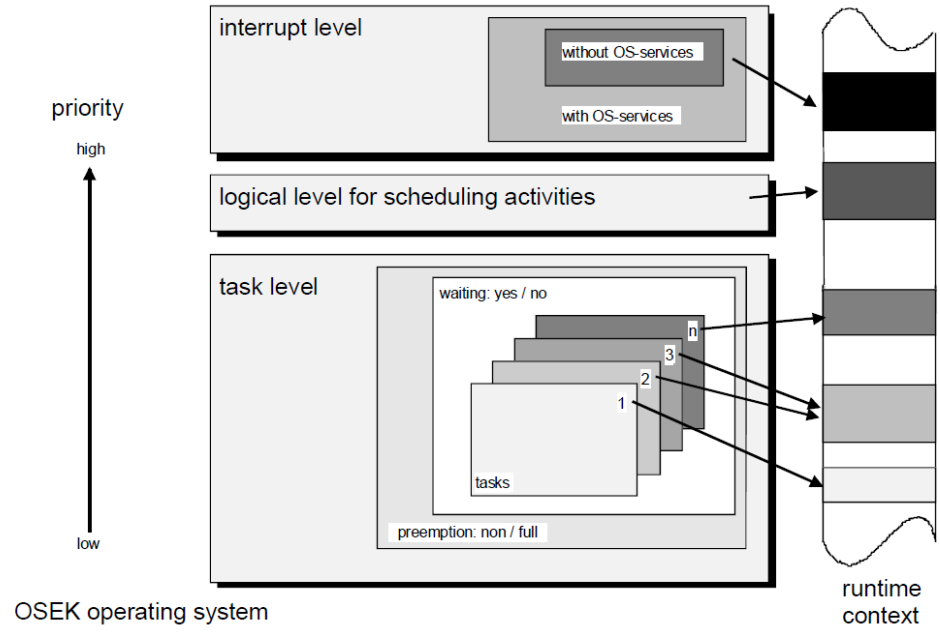
- Ρουτίνες εξυπηρέτησης σημάτων διακοπής που τις διαχειρίζεται το λειτουργικό σύστημα
- Εργασίες (βασικές και εκτεταμένες)

Οι πόροι υλικού από μια μονάδα ελέγχου μπορούν να διαχειρίζονται με τις υπηρεσίες του λειτουργικού συστήματος. Αυτές οι υπηρεσίες του λειτουργικού συστήματος καλούνται από μία μοναδική διεπαφή είτε από το πρόγραμμα εφαρμογή είτε από το εσωτερικό του ιδίου του λειτουργικού συστήματος.

Το OSEK ορίζει τρία επίπεδα επεξεργασίας:

- Επίπεδο σημάτων διακοπής.
- Λογικό επίπεδο για χρονοπρογραμματισμό.
- Επίπεδο εργασιών.

Μέσα στο επίπεδο εργασιών τα έργα χρονοπρογραμματίζονται ανάλογα με την προτεραιότητα που τους έχει αποδώσει ο χρήσης με προεκτόπιση, ή χωρίς προεκτόπιση ή μεικτά. Το απαιτούμενο περιβάλλον εκτέλεσης δεσμεύεται στις αρχές του χρόνου εκτέλεσης και απελευθερώνεται και πάλι με μιας όταν το έργο έχει τελειώσει.



Σχήμα 4.14 Το λειτουργικό σύστημα OSEK.

Έχουν καθοριστεί οι ακόλουθοι κανόνες προτεραιότητας:

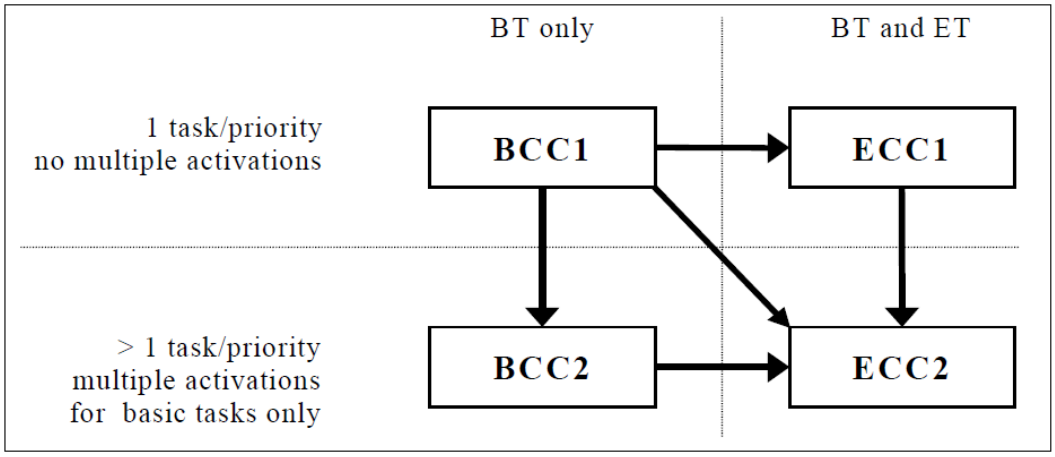
- Τα σήματα διακοπών έχουν προτεραιότητα σε σχέση με τις εργασίες.
- Το επίπεδο επεξεργασίας διακοπών αποτελείται από ένα ή περισσότερα επίπεδα προτεραιότητας σημάτων διακοπής.
- Στις ρουτίνες εξυπηρέτησης διακοπών έχουν αποδοθεί στατικά επίπεδα προτεραιότητας.
- Η ανάθεση προτεραιοτήτων σε ρουτίνες εξυπηρέτησης διακοπών εξαρτάται από την υλοποίηση και την αρχιτεκτονική του υλικού
- Για τις προτεραιότητες εργασιών και των πόρων μεγαλύτερο αριθμοί οροφή-προτεραιότητας αναφέρονται σε υψηλότερες προτεραιότητες.
- Η προτεραιότητα της εργασίας είναι στατικά καθορισμένη ν από το χρήστη.

Τα επίπεδα επεξεργασίας ορίζονται για την διεκπεραίωση των εργασιών και των ρουτινών εξυπηρέτησης διακοπών σαν μια σειρά από διαδοχικές τιμές. Η αντιστοίχιση των προτεραιοτήτων του λειτουργικού συστήματος με τις προτεραιότητες του υλικού γίνεται κατά την υλοποίηση.

Η ανάθεση αποτελεί προτεραιότητας για τον χρονοπρογραμματισμό είναι μόνο μια λογική έννοια η οποία μπορεί να υλοποιηθεί χωρίς την άμεση χρήση προτεραιότητας. Επιπλέον το OSEK δεν θεσπίζει κανένα κανόνα σχετικά με τη σχέση των προτεραιοτήτων εργασιών και τα επίπεδα σημάτων διακοπής του υλικού μιας συγκεκριμένης αρχιτεκτονικής μικροεπεξεργαστών.

Οι διάφορες απαιτήσεις των εφαρμογών λογισμικού από το σύστημα και οι διαφορετικές δυνατότητες ενός συγκεκριμένου συστήματος (π.χ. επεξεργαστής, μνήμη) ζητούν διαφορετικά χαρακτηριστικά από ένα λειτουργικό σύστημα. Αυτά τα λειτουργικά χαρακτηριστικά του συστήματος περιγράφονται στο OSEK σαν κατηγορίες συμμόρφωσης. Συνοπτικά έχουμε τις ακόλουθες κατηγορίες:

- BCC1 (μόνο βασικές εργασίες, με μία αίτηση ενεργοποίησης ανά εργασία και μία εργασία ανά προτεραιότητα και όλες οι εργασίες έχουν διαφορετικές προτεραιότητες).
- BCC2 (όπως το BCC1και περισσότερες από μία εργασία ανά προτεραιότητα είναι δυνατό και πολλαπλές ενεργοποιήσεις ανά εργασία επιτρέπονται).
- ECC1 (όπως το BCC1, καθώς και εκτεταμένες εργασίες).
- ECC2 (όπως ECC1, καθώς και περισσότερες από μία εργασία ανά προτεραιότητα είναι δυνατό και πολλαπλές αιτήσεις ενεργοποίησης ανά εργασία που επιτρέπονται από τις βασικές εργασίες).



Σχήμα 4.15 Κατηγορίες συμμόρφωσης με συμβατότητα προς τα πάνω.

Η φορητότητα των εφαρμογών μπορεί να θεωρηθεί ότι ισχύει εφόσον δεν υπερβαίνουν τις ελάχιστες απαιτήσεις. Οι ελάχιστες απαιτήσεις για τις κατηγορίες συμμόρφωση φαίνεται στο σχήμα 4.16.

	BCC1	BCC2	ECC1	ECC2
Multiple requesting of task activation	no	yes	BT[3]: no ET: no	BT: yes ET: no
Number of tasks which are not in the *suspended* state	8		16 (any combination of BT/ET)	
More than one task per priority	no	yes	no (both BT/ET)	yes (both BT/ET)
Number of events per task	—		8	
Number of task priorities	8		16	
Resources	RES_SCHEDULER	8 (including RES_SCHEDULER)		
Internal resources	2			
Alarm	1			
Application Mode	1			

Σχήμα 4.16 Ελάχιστες προδιαγραφές κατηγοριών συμμόρφωσης.

Σχέση μεταξύ OSEK OS και OSEKtime OS

Το OSEKtime OS είναι ένα λειτουργικό σύστημα ειδικά προσαρμοσμένο στις ανάγκες των αρχιτεκτονικών που ενεργοποιούνται από χρονικά συμβάντα. Επιτρέπεται το OSEK OS να συνυπάρχει με το OSEKtime OS. Το OSEKtime διαθέτει το χρόνο που είναι σε αδράνεια να χρησιμοποιείται από OSEK. Το Οι διακοπές και οι διεργασίες του OSEK έχουν χαμηλότερη προτεραιότητα σε σχέση με παρόμοιες οντότητες του OSEKtime OS.

Οι διασυνδέσεις OSEK, και ο ορισμός των κλήσεων συστήματος, δεν αλλάζουν αν το OSEK συνυπάρχει με OSEKtime. Υπάρχουν μικρές εξαιρέσεις στην εκκίνηση του συστήματος και το κλείσιμο του που οφείλεται στο γεγονός ότι το OSEKtime είναι υπεύθυνο για όλο το σύστημα ενώ το OSEK είναι μόνο τοπικά. Αυτές οι αποκλίσεις αναφέρονται ρητά στις προδιαγραφές.

Διαχείριση διεργασιών.

Το λειτουργικό σύστημα OSEK έχει δύο διαφορετικές διεργασίες:

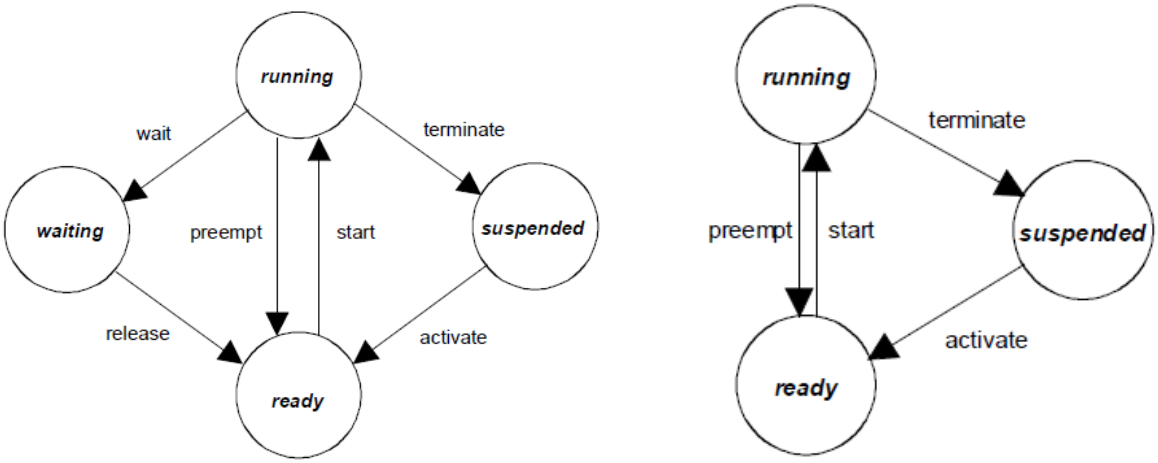
- βασική διεργασία
- εκτεταμένη διεργασία.

Οι βασικές διεργασίες απελευθερώνουν τον επεξεργαστή, αν

- τερματίσουν,
- το OSEK φορτώσει μία διεργασία υψηλότερης προτεραιότητας, ή
- προκύπτει μια διακοπή, που οδηγεί τον επεξεργαστή στην εκτέλεση της ρουτίνας εξυπηρέτησης της διακοπής(ISR).

Εκτεταμένη Εργασίες

Οι Εκτεταμένη διεργασίες διακρίνονται από τις βασικές με το να επιτρέπεται να χρησιμοποιούν κλήση WaitEvent του λειτουργικού συστήματος, που μπορεί να οδηγήσει σε κατάσταση αναμονής. Η κατάσταση αναμονής επιτρέπει τον επεξεργαστή να αποδοθεί σε μία διεργασία χαμηλότερης προτεραιότητας, χωρίς να χρειάζεται να τερματίσει τη λειτουργία της η εκτελούμενη εκτεταμένη διεργασία. Από την πλευρά του λειτουργικού συστήματος, η διαχείριση των εκτεταμένων διεργασιών είναι πιο σύνθετη από τη διαχείριση των βασικών και απαιτεί περισσότερους πόρους του συστήματος.



Σχήμα 4.17 Καταστάσεις και μεταβάσεις για εκτεταμένες και βασικές διεργασίες.

Μία εκτεταμένη διεργασία μπορεί να βρίσκεται σε μία από τέσσερις καταστάσεις:, running, ready, waiting και suspended. Οι καταστάσεις και οι μεταβάσεις φαίνονται στο σχήμα 4.17α. Μία διεργασία τερματίζεται από μόνη της. Δεν είναι δυνατή, για να είναι απλό το λειτουργικό σύστημα, η μετάβαση από την κατάσταση suspend απ' ευθείας στην κατάσταση wait.

Μία βασική διεργασία μπορεί να βρίσκεται σε μία από τρεις καταστάσεις:, running, ready, και suspended. Οι καταστάσεις και οι μεταβάσεις φαίνονται στο σχήμα 4.17β.

Επειδή οι βασικές διεργασίες δεν έχουν κατάσταση wait, τα μόνα σημεία συγχρονισμού που υπάρχουν είναι στην αρχή και στο τέλος της διεργασίας. Τμήματα μίας εφαρμογής με εσωτερική σημεία συγχρονισμού υλοποιούνται με περισσότερες από μία βασικές διεργασίες. Ένα πλεονέκτημα των βασικών διεργασιών είναι οι μικρές απαιτήσεις τους για περιβάλλον εκτέλεσης. Οι εκτεταμένες διεργασίες έχουν το πλεονέκτημα του ότι μπορούν να χειριστούν μια εργασίας σαν μία ενιαία διεργασία, ανεξάρτητα από τη μέθοδο συγχρονισμού που είναι ενεργός. Οι εκτεταμένες διεργασίες περιλαμβάνουν επίσης περισσότερα σημεία συγχρονισμού από τις βασικές.

Ενεργοποίηση μιας διεργασίας.
Μία διεργασία ενεργοποιείται με τις υπηρεσίες ActivateTask ή ChainTask του λειτουργικού συστήματος. Μετά την ενεργοποίηση της η διεργασία είναι έτοιμη για εκτελέσει από την πρώτη γραμμή.

Το λειτουργικό σύστημα OSEK δεν υποστηρίζει το πέρασμα παραμέτρων κατά την έναρξη μιας διεργασίας. Αυτές οι παράμετροι θα πρέπει να περάσουν είτε μέσω διαβίβασης μηνύματος είτε με καθολικές μεταβλητές.

Προτεραιότητες διεργασιών.
Ο χρονοπρογραμματιστής αποφασίζει με βάση την προτεραιότητα της διεργασίας ποια είναι η επόμενη από τις έτοιμες διεργασίες που πρόκειται να μεταφερθεί στη κατάσταση εκτέλεσης.

Η τιμή 0 ορίζεται ως η χαμηλότερη προτεραιότητα μιας διεργασίας. Κατά συνέπεια μεγαλύτεροι αριθμοί καθορίζουν υψηλότερες προτεραιότητες. Για να ενισχυθεί η αποτελεσματικότητα, δεν υποστηρίζεται δυναμική διαχείριση προτεραιοτήτων. Κατά συνέπεια, η προτεραιότητα της διεργασίας ορίζεται στατικά, δηλαδή ο χρήστης δεν μπορεί να την αλλάξει κατά την εκτέλεσή της. Ωστόσο, σε ειδικές περιπτώσεις, το λειτουργικό σύστημα μπορεί να δει μια διεργασία με μια καθορισμένη μεγαλύτερη προτεραιότητα (πρωτόκολλο PCP). Στις κλάσεις συμμόρφωση BCC2 και ECC2, υποστηρίζονται πολλές διεργασίες με την ίδια προτεραιότητα.

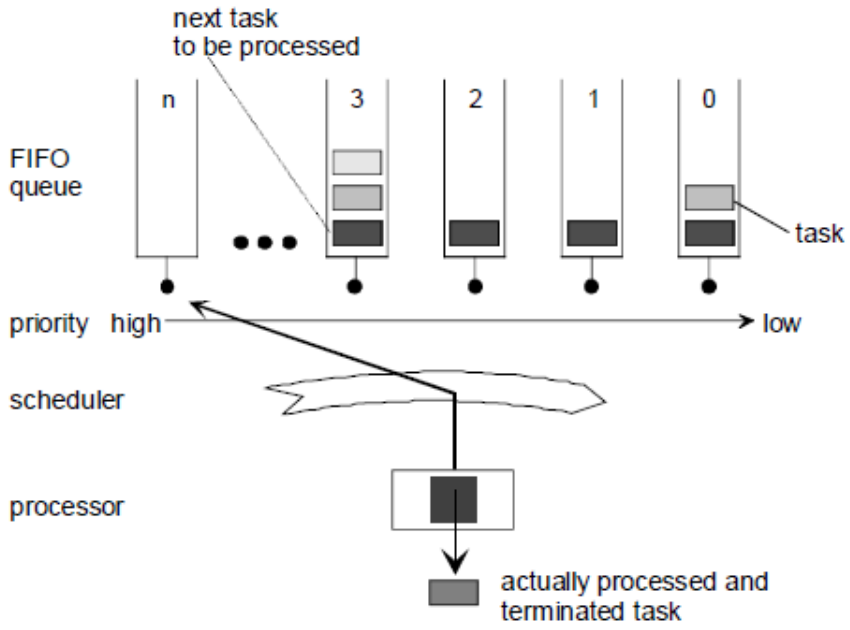
Οι διεργασίες στο ίδιο επίπεδο προτεραιότητας εκκινούν ανάλογα με τη σειρά ενεργοποίησής τους. Στις εκτεταμένες διεργασίες η διεργασία που είναι σε κατάσταση αναμονής δεν μπλοκάρει την έναρξη των άλλων διεργασιών της ίδιας προτεραιότητας.

Μια προεκτοπισμένη διεργασία θεωρείται ότι είναι η πρώτη (παλαιότερη) εργασία στη λίστα των έτοιμων διεργασιών με την τρέχουσα προτεραιότητά της. Μια διεργασία που επέστρεψε από την κατάσταση αναμονής αντιμετωπίζεται ως η τελευταία (νεότερη) στην ουρά έτοιμων διεργασιών της προτεραιότητάς της.

Το σχήμα 4.18 δείχνει ένα παράδειγμα εφαρμογής του χρονοπρογραμματισμού που χρησιμοποιείται σε κάθε επίπεδο προτεραιότητας.

Τα βασικά βήματα που είναι απαραίτητα για να προσδιοριστεί η επόμενη διεργασία για αποστολή προς επεξεργασία είναι:

- Η αναζήτηση όλων των έτοιμων –εκτελούμενων διεργασιών.
- Ο χρονοπρογραμματιστής ορίζει το σύνολο των εργασιών με την υψηλότερη προτεραιότητα.
- Στο σύνολο των εργασιών ύψιστης προτεραιότητας βρίσκει την αρχαιότερη διεργασία.



Σχήμα 4.18 Επίπεδα προτεραιότητας για χρονοπρογραμματισμό.

Η πολιτική προεκτόπισης μπορεί να είναι πλήρως προεκτοπιστική ή μη προεκτοπιστική. Ο προγραμματιστής του λογισμικού ή της δημιουργίας του συνολικού συστήματος καθορίζει την ακολουθία εκτέλεση της διεργασίας με τη διαμόρφωση των προτεραιοτήτων εργασίας και την ανάθεση της προεκτόπισης ως χαρακτηριστικό της διεργασίας.

Ο τύπος διεργασίας (βασική ή εκτεταμένη) είναι ανεξάρτητος από τον τύπο του χρονοπρογραμματισμού της εργασίας (preemptable ή μη preemptable). Ένα πλήρως προεκτοπιστκό σύστημα μπορεί να περιέχει βασικές διεργασίες και ένα μη προεκτοπιστικό σύστημα να έχει εκτεταμένες διεργασίες.

Πολλές εφαρμογές περιλαμβάνουν μόνο λίγες παράλληλες διεργασίες με μεγάλο χρόνο εκτέλεσης, για το οποίο ένα πλήρως προεκτοπιστικό λειτουργικό σύστημα θα ήταν βολικό και πολλές σύντομες εργασίες με καθορισμένη προθεσμία εκτέλεσης, όπου ο μη προεκτοπιστικός χρονοπρογραμματισμός θα ήταν πιο αποτελεσματικός. Η διαμόρφωση μικτής πολιτικής προεκτόπισης αναπτύχθηκε ως συμβιβαστική λύση. Αν εκτελείται μια υπηρεσία του λειτουργικού συστήματος, η προεκτόπιση και η εναλλαγή πλαισίου εργασίας μπορεί να αναβληθεί μέχρι την ολοκλήρωση της υπηρεσίας.

Διαχείριση σημάτων διακοπής.
Οι ρουτίνες εξυπηρέτησης σημάτων διακοπής χωρίζονται σε δύο κατηγορίες:

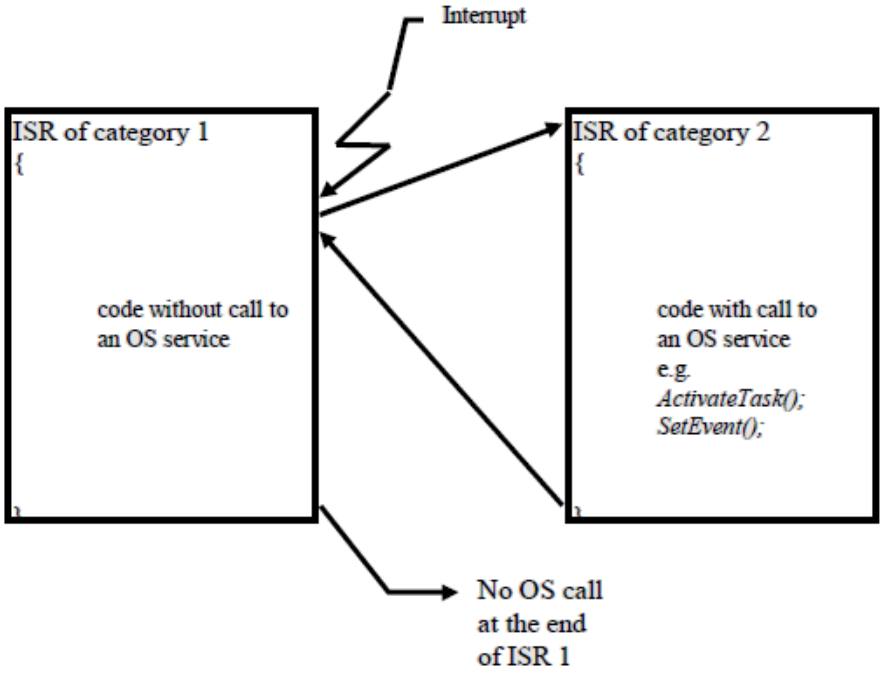
ISR category1. Δεν γίνεται χρήση των κλήσεων του συστήματος πλην των enable και disable interrupt. Μετά την ολοκλήρωση της ρουτίνας ISR συνεχίζεται η εκτέλεση της διεργασίας από το σημείο που είχε σταματήσει. Το σήμα διακοπής δεν επηρέασε τη διαχείριση των διεργασιών.

ISR category2. Το λειτουργικό σύστημα OSEK παρέχει ένα πλαίσιο-ISR για να προετοιμάσει ένα περιβάλλον εκτέλεσης για μια ειδική ρουτίνα του χρήστη. Κατά τη διάρκεια της δημιουργίας του συστήματος η ρουτίνα του χρήστη έχει αντιστοιχηθεί στο σήμα διακοπής.

Στο εσωτερικό της ρουτίνας εξυπηρέτησης ISR δεν μπορεί να πραγματοποιηθεί επαναχρονοπρογραμματισμός. Επαναχρονοπρογραμματισμός γίνεται μετά τη λήξη της ISR κατηγορία 2 εάν μία προεκτοπίσιμη διεργασία έχει διακοπεί και αν καμία άλλη διακοπή δεν είναι ενεργή. Η υλοποίηση εξασφαλίζει ότι οι εργασίες εκτελούνται σύμφωνα με τον

χρονοπρογραμματισμό του OSEK Για να επιτευχθεί αυτό η υλοποίηση μπορεί να βάλλει περιορισμούς σχετικά με τα επίπεδα προτεραιότητας διακοπών για ISR όλων των κατηγοριών ή και να προβεί σε ελέγχους κατά το χρόνο διαμόρφωσης του συστήματος. Ο μέγιστος αριθμός των προτεραιοτήτων σημάτων διακοπής εξαρτάται από τον ελεγκτή, καθώς τη σχετική υλοποίηση. Ο προγραμματισμός των διακοπών είναι στην πλευρά του υλικού και δεν προσδιορίζονται από το OSEK. Τα σήματα διακοπής έχουν προγραμματιστεί από το υλικό, ενώ οι χρονοπρογραμματιζόμενες διεργασίες από τον χρονοπρογραμματιστή. Όσον αφορά τα επίπεδα προτεραιότητας των διακοπών ενδέχεται να υπάρχουν περιορισμοί Σήματα διακοπής μπορεί να διακόψουν τις διεργασίες (προεκτοπίσιμες ή μη προεκτοπίσιμες). Εάν μια διεργασία έχει ενεργοποιηθεί από μια ρουτίνα εξυπηρέτησης σήματος διακοπής έχει χρονοπρογραμματιστεί μετά το πέρας όλων των ενεργών ρουτινών εξυπηρέτησης σημάτων διακοπής. Το OSEK προσφέρει γρήγορες λειτουργίες για να απενεργοποιήσετε όλα τα σήματα διακοπής και να απενεργοποιήσετε όλες τις διακοπές της κατηγορίας 2. Τυπική χρήση τους είναι η προστασία σύντομων κρίσιμων τμημάτων.

Δεδομένου ότι όλες οι διακοπές έχουν μεγαλύτερη προτεραιότητα από τις διεργασίες, η επεξεργασία των διακοπών λήγει πριν το σύστημα επιστρέψει στο επίπεδο διεργασιών. Αν μια ISR της κατηγορίας 2 διακόπτει μια ISR κατηγορίας 1, το σύστημα θα συνεχίσει την επεξεργασία του ISR1 μετά τον τερματισμό του ISR2. Έχοντας ενεργοποιημένες διεργασίες ή συμβάντα που έγιναν set από το επίπεδο ISR2 το λειτουργικό σύστημα δεν καλείται μετά τη λήξη της ISR1, προκειμένου να εκτελέσει νέο χρονοπρογραμματισμό. Επειδή δε οι ISR κατηγορίας 1 δεν λειτουργούν υπό τον έλεγχο του λειτουργικού συστήματος το λειτουργικό σύστημα δεν έχει τη δυνατότητα να εκτελέσει νέο χρονοπρογραμματισμό όταν η ISR τερματίζει. Έτσι οι δραστηριότητες που αντιστοιχούν στις κλήσεις του λειτουργικού συστήματος και στις ISR2 καθυστερούν μέχρι το επόμενο σημείο χρονοπρογραμματισμού.



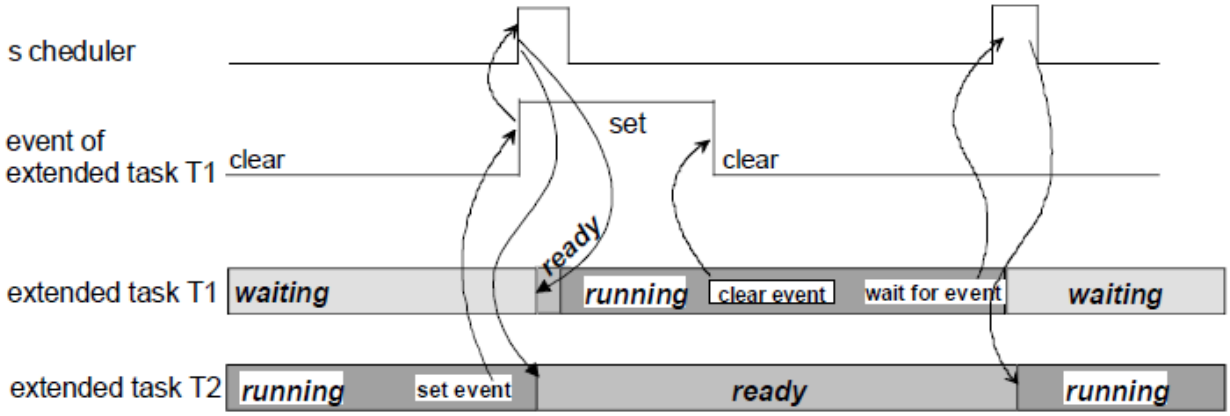
Σχήμα 4.19 Εμφωλευμένα διακοπές.

Για να μην συμβούν τα προβλήματα που αναφέρθηκαν παραπάνω, κάθε σύστημα πρέπει να δημιουργήσει κανόνες για την αποφυγή αυτών των προβλημάτων. Για μέγιστη φορητότητα μίας εφαρμογή, ένας εύκολος κανόνας που πάντα δουλεύει είναι: όλα τα σήματα διακοπών κατηγορίας 1 πρέπει να έχουν μεγαλύτερη ή ίση προτεραιότητα υλικού σε σχέση με διακοπές της κατηγορίας 2.

Μηχανισμός συμβάντων.

Ο μηχανισμός συμβάντων είναι ένα μέσο συγχρονισμού, παρέχεται μόνο για τις εκτεταμένες διεργασίες και αρχίζει τις μεταβάσεις μίας διεργασίας από και προς την κατάσταση αναμονής.

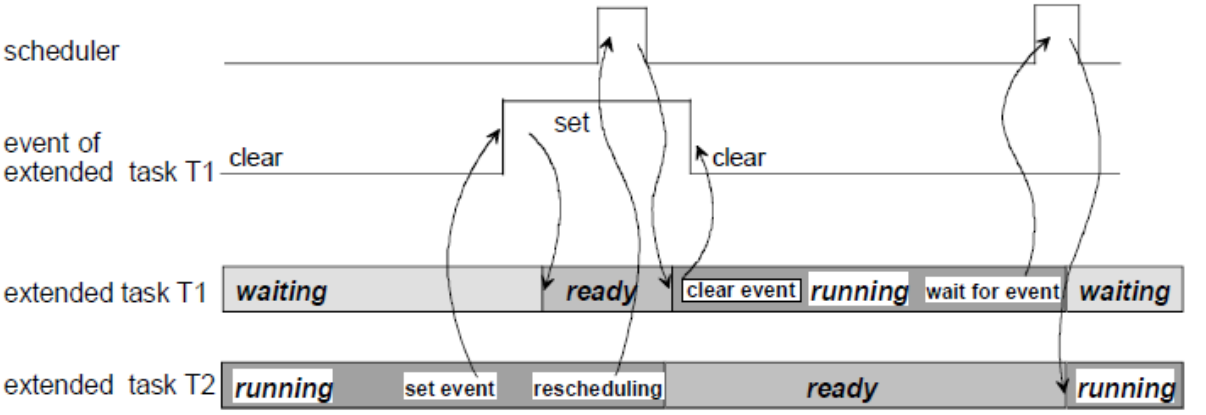
Τα συμβάντα είναι αντικείμενα που διαχειρίζεται το λειτουργικό σύστημα. Δεν είναι ανεξάρτητα αντικείμενα, έχουν αποδοθεί σε εκτεταμένες διεργασίες. Κάθε εκτεταμένη διεργασία έχει μια καθορισμένη σειρά συμβάντων. Η διεργασία αυτή καλείται ιδιοκτήτης αυτών των συμβάντων. Ένα μεμονωμένο συμβάν προσδιορίζεται από τον ιδιοκτήτη του και το όνομά του. Όταν ενεργοποιείτε μία εκτεταμένη διεργασία, αυτά καθαρίζονται από το λειτουργικό σύστημα. Τα συμβάντα μπορούν να χρησιμοποιηθούν για να μεταφέρουν δυαδική πληροφορία στην εκτεταμένη διεργασία που τους έχει αντιστοιχηθεί. Το νόημα των συμβάντων καθορίζεται από την εφαρμογή, π.χ. σηματοδότηση της λήξης ενός χρονικού διαστήματος, η διαθεσιμότητα ενός πόρου, η λήψη μιας πληροφορίας, κλπ.



Σχήμα 4.20 Συγχρονισμό εκτεταμένων διεργασιών με συμβάντα και προεκτόπιση.

Ο δέκτης ενός συμβάντος είναι πάντα μια εκτεταμένη διεργασία. Δεν είναι δυνατόν μια ρουτίνα εξυπηρέτησης διακοπής ή μια βασική διεργασία να περιμένει για ένα συμβάν. Μια εκτεταμένη διεργασία σε κατάσταση αναμονής απελευθερώνεται σε κατάσταση ετοιμότητας όταν τουλάχιστον ένα συμβάν που περιμένει, συμβεί.

Το σχήμα 4.20 απεικονίζει τον συγχρονισμό εκτεταμένων διεργασιών με συμβάντα στην περίπτωση που υπάρχει προεκτοπιστικός χρονοπρογραμματισμός.



Σχήμα 4.21 Συγχρονισμό εκτεταμένων διεργασιών με συμβάντα χωρίς προεκτόπιση.

Το σχήμα 4.21απεικονίζει τον συγχρονισμό εκτεταμένων διεργασιών με συμβάντα στην περίπτωση που υπάρχει μη προεκτοπιστικός χρονοπρογραμματισμός.

Διαχείριση πόρων.

Η διαχείριση των πόρων χρησιμοποιείται για να συντονίσει την ταυτόχρονη πρόσβαση πολλών διεργασιών με διαφορετικές προτεραιότητες σε κοινόχρηστους πόρους, π.χ. μία περιοχή μνήμης ή κάποιο υλικό. Η διαχείριση των πόρων είναι υποχρεωτική για όλες τις κατηγορίες συμμόρφωση. Προαιρετικά μπορεί να επεκταθεί ώστε να συντονίσει την ταυτόχρονη πρόσβαση των διεργασιών και των ρουτινών εξυπηρέτησης διακοπών.

Η διαχείριση των πόρων εξασφαλίζει ότι: δύο εργασίες δεν μπορούν να καταλάβουν την ίδια πηγή ταυτόχρονα, δεν μπορεί να συμβεί αναστροφή προτεραιότητα, δεν εμφανίζονται αδιέξοδα κατά τη χρήση αυτών των πόρων και η πρόσβαση σε κάποιο πόρο δεν οδηγεί σε κατάσταση αναμονής.

Εάν η διαχείριση των πόρων επεκταθεί και στο επίπεδο σημάτων διακοπής διαβεβαιώνει επιπλέον ότι δύο διεργασίες ή ρουτίνες εξυπηρέτησης διακοπής δεν μπορούν να καταλάβουν την ίδια πηγή ταυτόχρονα.

Το OSEK OS καθορίζει την προτεραιότητα πρωτοκόλλου οροφής OSEK. Κατά συνέπεια, δεν εμφανίζεται κατάσταση στην οποία μια εργασία ή ένα σήμα διακοπής να προσπαθούν να αποκτήσουν πρόσβαση σε ένα δεσμευμένο πόρο.

Εάν η έννοια των πόρων χρησιμοποιείται για το συντονισμό των διεργασιών και των διακοπών το λειτουργικό σύστημα OSEK διασφαλίζει ότι η ρουτίνα εξυπηρέτησης της διακοπής εκτελείται μόνο όταν έχει όλους τους απαραίτητους πόρους που θα χρειαστεί διαθέσιμους. Το OSEK απαγορεύει αυστηρά την εμφωλευμένη πρόσβαση στον ίδιο πόρο. Σε σπάνιες περιπτώσεις που είναι απαραίτητο, συνιστάται η χρήση μιας δεύτερης πηγής με την ίδια συμπεριφορά με την πρώτη πηγή. Η γλώσσα OIL στηρίζει τον ορισμό πόρων με την ίδια πανομοιότυπη συμπεριφορά («συνδεδεμένοι πόροι»).

Στην περίπτωση πολλαπλής απασχόλησης των πόρων μέσα στην ίδια διεργασία, ο χρήστης πρέπει να ζητήσει και να απελευθερώσει τους πόρους ακολουθώντας την αρχή LIFO.

Διαχείριση σφαλμάτων.

Μια υπηρεσία σφάλματος παρέχεται για να χειριστεί τα προσωρινά και τα μόνιμα λάθη μέσα στο λειτουργικό σύστημα OSEK. Το βασικό πλαίσιο είναι προκαθορισμένο και συμπληρώνεται από τον χρήστη. Αυτό δίνει στο χρήστη τη δυνατότητα επιλογής συγκεντρωτικής ή αποκεντρωμένης διαχείρισης λαθών.

Δύο διαφορετικά είδη λαθών αναγνωρίζονται:

- Σφάλματα εφαρμογών Το λειτουργικό σύστημα δεν θα μπορούσε να εκτελέσει την αιτούμενη υπηρεσία σωστά, αλλά υποθέτει την ορθότητα των εσωτερικών δεδομένων του. Στην περίπτωση αυτή καλείται η κεντρική διαχείριση σφαλμάτων. Επιπλέον, το λειτουργικό σύστημα επιστρέφει το σφάλμα και τις πληροφορίες κατάστασης για αποκεντρωμένη επεξεργασία σφαλμάτων. Εξαρτάται από το χρήστη να αποφασίσει τι πρέπει να κάνετε ανάλογα με το σφάλμα.
- Τα ανεπανόρθωτα σφάλματα. Το λειτουργικό σύστημα δεν μπορεί πλέον να υποθέσει την ορθότητα των εσωτερικών δεδομένων της. Στην περίπτωση αυτή το λειτουργικό σύστημα καλεί την κεντρική διακοπή λειτουργίας του συστήματος.

Όλες οι υπηρεσίες αυτές έχουν αντιστοιχηθεί με μια παράμετρο που καθορίζει το σφάλμα. Το λειτουργικό σύστημα OSEK προσφέρει δύο επίπεδα για έλεγχο σφαλμάτων, την τυπική κατάσταση και την εκτεταμένη κατάσταση. Η τιμή επιστροφής της OSEK API υπηρεσίας έχει προτεραιότητα σε σχέση με τις παραμέτρους εξόδου. Αν μια υπηρεσία API επιστρέφει ένα σφάλμα, οι τιμές των παραμέτρων εξόδου είναι απροσδιόριστες.

4.2.3 QNX (www.qnx.com).

Η πρώτη έκδοση του QNX κυκλοφόρησε το 1981. Η τελευταία έκδοση είναι η QNX Neutrino OS 6.5[32]. Κάθε γενιά κληρονομούσε την εμπειρία της προηγούμενης.

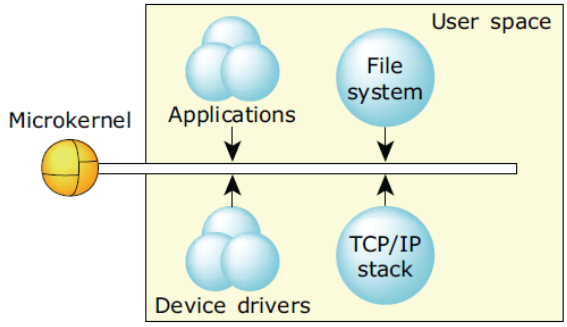
Η κύρια ευθύνη του λειτουργικού συστήματος είναι να διαχειρίζεται τους πόρους ενός υπολογιστή. Όλες οι δραστηριότητες στο σύστημα θα πρέπει να λειτουργούν μαζί απρόσκοπτα και με τη μεγαλύτερη δυνατή διαφάνεια. Κάποια περιβάλλοντα απαιτούν πιο αυστηρή διαχείριση των πόρων και πιο αυστηρό χρονοπρογραμματισμό. Για παράδειγμα, οι εφαρμογές πραγματικού χρόνου, εξαρτώνται από το λειτουργικό σύστημα για να χειρίζεται πολλαπλά γεγονότα και να διασφαλίζει ότι το σύστημα ανταποκρίνεται σε αυτά στο όρια των προθεσμιών που έχουν τεθεί. Όσο πιο γρήγορα ανταποκρίνεται το λειτουργικό σύστημα, τόσο περισσότερο χρόνο έχει μία εφαρμογή πραγματικού χρόνου να τήρηση τις προθεσμίες της.

Το QNX Neutrino είναι κατάλληλο για ενσωματωμένες εφαρμογές πραγματικού χρόνου γιατί μπορεί να κλιμακωθεί σε πολύ μικρά μεγέθη. Παρέχει πολυδιεργασία, νήματα, προεκτοπιστικό χρονοπρογραμματισμό οδηγούμενο από προτεραιότητες και γρήγορη μεταγωγή περιβάλλοντος. Όλα είναι βασικά συστατικά ενός ενσωματωμένου συστήματος πραγματικού χρόνου. Επιπλέον είναι συμβατό με το πρότυπο POSIX.

Οι προγραμματιστές μπορούν εύκολα να προσαρμόσουν το λειτουργικό σύστημα QNX Neutrino να ανταποκρίνεται στις ανάγκες της εφαρμογής τους. Από τη διαμόρφωση ενός μικροπυρήνα με μερικές μικρές ενότητες μόνο, μέχρι ένα πλήρως εξοπλισμένο δίκτυο σε επίπεδο συστήματος για να εξυπηρετήσει εκατοντάδες χρήστες, ο προγραμματιστής έχει τη δυνατότητα να ρυθμίσει το σύστημα να χρησιμοποιεί μόνο τους πόρους που πραγματικά χρειάζεται.

Το QNX Neutrino ακολουθεί δύο θεμελιώδεις αρχές:

- αρχιτεκτονική μικροπυρήνα
- η διαδιεργασιακή επικοινωνία γίνεται μέσω μηνυμάτων.

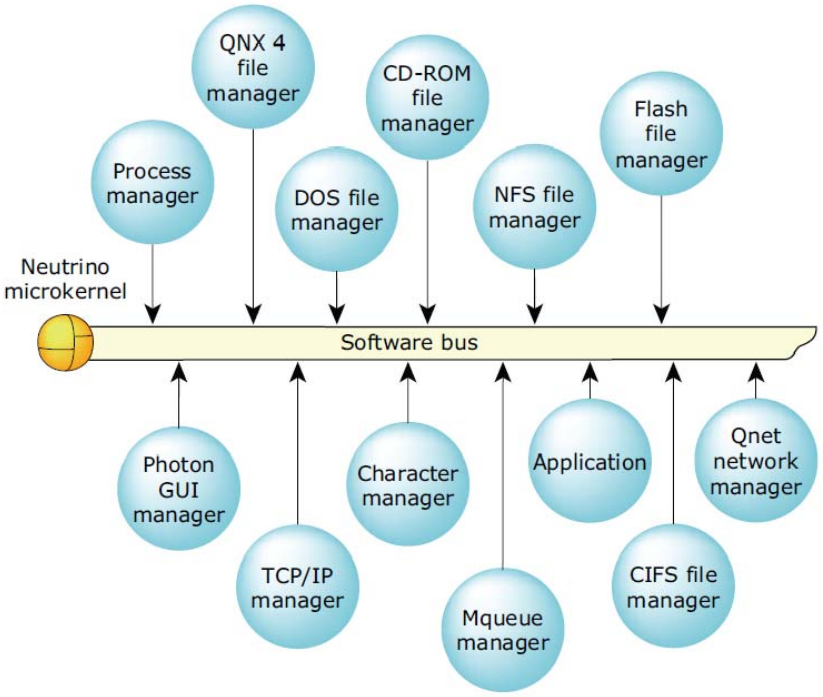


Σχήμα 4.22 Η δομή μικροπυρήνα παρέχει προστασία μνήμης.

Ο πραγματικός στόχος για τον σχεδιασμό ενός λειτουργικού συστήματος με μικροπυρήνα είναι η επεκτασιμότητα, το μέγεθος προκύπτει σαν μια παρενέργεια. Ο μικροπυρήνας διαφέρει από τις μονολιθικές υλοποιήσεις στον τρόπο που οι διαδιεργασιακές υπηρεσίες επικοινωνίας χρησιμοποιούνται για να αυξήσουν την λειτουργικότητα του πυρήνα με προσθήκη, διεργασιών παροχής υπηρεσιών. Δεδομένου ότι το λειτουργικό σύστημα υλοποιείται ως μια ομάδα συνεργαζόμενων διεργασιών που διαχειρίζεται τον μικροπυρήνα, οι διεργασίες που γράφονται από τον χρήστη μπορεί να χρησιμεύσουν τόσο ως εφαρμογές όσο και ως διεργασίες που επεκτείνουν την λειτουργικότητα του συστήματος. Το ίδιο το λειτουργικό σύστημα γίνεται ανοικτό και εύκολα επεκτάσιμη. Επιπλέον, οι επεκτάσεις από το χρήστη δεν θα επηρεάσουν την αξιοπιστία του πυρήνα του λειτουρ-

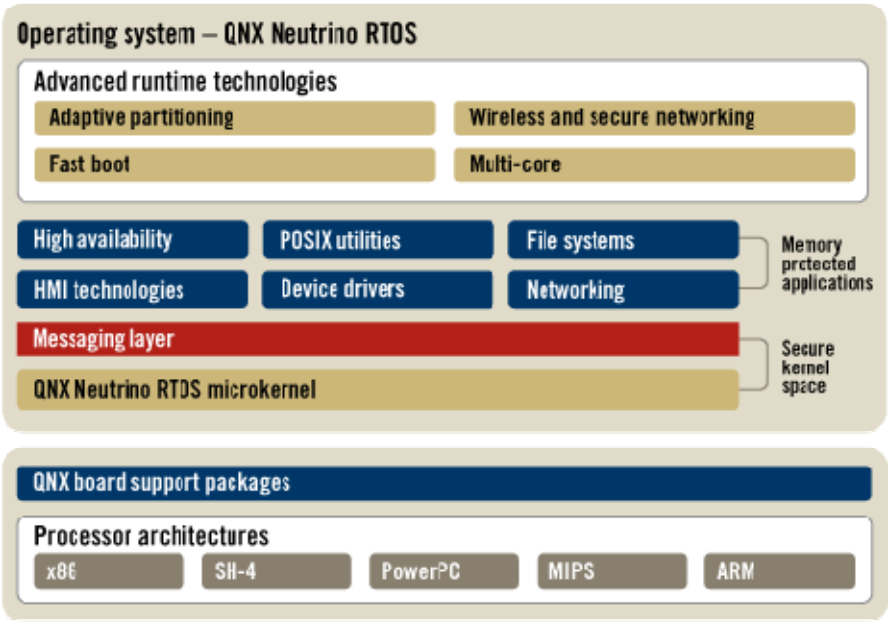
γικού συστήματος. Μια δυσκολία για την εφαρμογή του προτύπου POSIX 1003.1 είναι ότι το περιβάλλον εκτέλεσης είναι συνήθως μία και μόνη-διεργασία με πολυνηματικό μοντέλο, χωρίς προστασία μνήμης μεταξύ των νημάτων. Αυτό είναι ένα υποσύνολο του πολυδιεργασιακού μοντέλου που POSIX. Το QNX Neutrino χρησιμοποιεί μία MMU για να υλοποιήσει πλήρως το διεργασιακό μοντέλο του POSIX σε ένα προστατευμένο περιβάλλον.

Η αρχιτεκτονική του QNX απεικονίζεται στα σχήματα 4.23 και 4.24.



Σχήμα 4.23 Η αρχιτεκτονική του QNX.

Το λειτουργικό σύστημα QNX Neutrino αποτελείται από ένα μικρό μικροπυρήνα που διαχειρίζεται μια ομάδα συνεργαζομένων διεργασιών. Λειτουργεί σαν μία αρτηρία λογισμικού που επιτρέπει δυναμικά την φόρτωση και την εκφόρτωση μονάδων λογισμικού κάθε φορά που χρειάζονται.

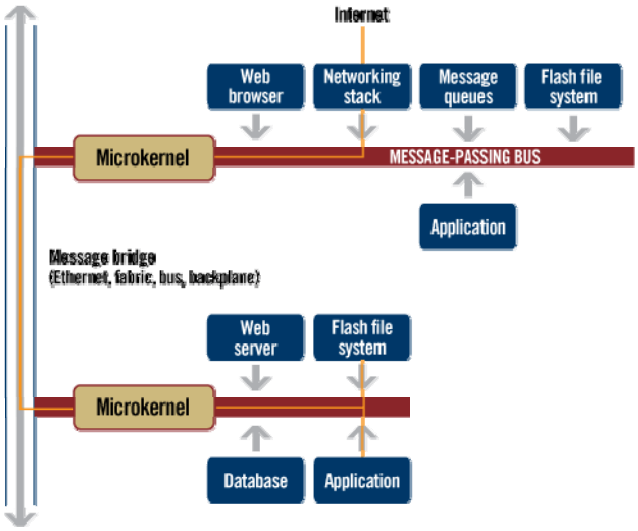


Σχήμα 4.24 Το QNX[34].

Ο πυρήνας έχει μικρό μέγεθος και υποστηρίζει λίγες βασικές υπηρεσίες:

- thread services.
- signal services.
- message-passing services.
- synchronization services.
- scheduling services.
- timer services.
- process management services.

Όλες οι υπηρεσίες του λειτουργικού συστήματος, εκτός αυτών που παρέχονται υποχρεωτικά από το μικροπυρήνα, διακινούνται μέσω προτύπων διεργασιών. Ένα μεγάλο σύστημα μπορεί να περιλαμβάνει σύστημα διαχείρισης αρχείων, διαχειριστή συσκευών ρυθμού χαρακτήρα, γραφική διεπαφή χρήστη (Photon), εγγενή διαχειριστή δικτύου, TCP/IP και άλλα.

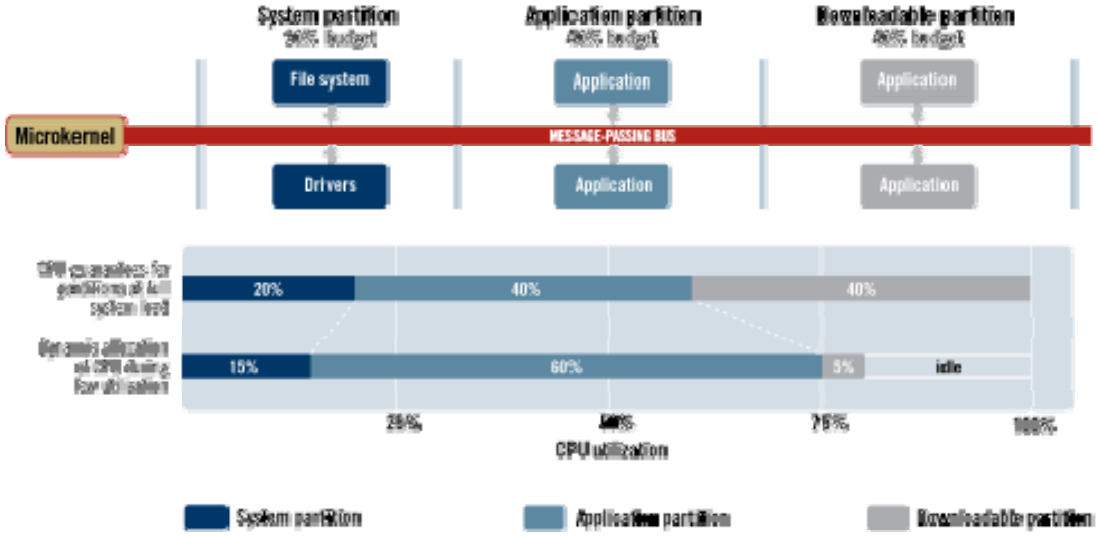


Σχήμα 4.25 Transparent Distributed Processing[34].

Οι οδηγοί συσκευών επιτρέπει το λειτουργικό σύστημα και τα προγράμματα εφαρμογής να κάνουν χρήση του υποκείμενου υλικού. Στο QNX, σε αντίθεση με τα άλλα λειτουργικά συστήματα που απαιτούν οι οδηγοί συσκευών να είναι στενά συνδεδεμένοι με το ίδιο το λειτουργικό σύστημα, μπορεί να ξεκινήσουν και να σταματήσουν σαν πρότυπες διεργασίες. Έτσι η προσθήκη τους δεν επηρεάζει κανένα άλλο μέρος του λειτουργικού συστήματος. Μπορούν να αναπτυχθούν και να εκσφαλματωθούν όπως και οποιαδήποτε άλλη εφαρμογή.

Το QNX Neutrino ενσωματώνει το σύνολο του δικτύου σε ένα ενιαίο, ομοιογενές σύνολο πόρων. Κάθε νήμα σε κάθε μηχάνημα στο δίκτυο μπορεί να κάνει άμεσα χρήση των πόρων σε κάθε άλλο μηχάνημα. Από την πλευρά της εφαρμογής, δεν υπάρχει καμία διαφορά μεταξύ ενός τοπικού ή απομακρυσμένου πόρου δεν απαιτούνται ειδικές ρυθμίσεις για τις εφαρμογές ώστε να τους επιτρέπεται να κάνουν χρήση των απομακρυσμένων πόρων. Οι χρήστες μπορούν να έχουν πρόσβαση στα αρχεία οπουδήποτε στο δίκτυο, να επωφεληθούν από οποιαδήποτε περιφερειακή συσκευής και την εκτέλεση εφαρμογών σε οποιοδήποτε υπολογιστή στο δίκτυο (με την προϋπόθεση ότι έχουν την εξουσιοδότηση γι' αυτό). Οι διεργασίες μπορούν να επικοινωνούν με τον ίδιο τρόπο οπουδήποτε σε όλο το δίκτυο. Στην καρδιά του QNX Neutrino το μητρικό πρωτόκολλο δικτύωσης είναι το Qnet που έχει αναπτυχθεί ως ένα δίκτυο από στενά συνδεδεμένες αξιόπιστες μηχανές. Το Qnet επιτρέπει σε αυτές τις μηχανές να μοιράζονται τους πόρους τους αποδοτικά με μικρή επιβάρυνση.

Ένα άλλο σημαντικό χαρακτηριστικό του QNX είναι η προσαρμοστική τμηματοποίηση που είναι μια απλή, αξιόπιστη λύση για τον συστήματα με εντατική χρήση επεξεργαστή. Διασφαλίζει ότι ποτέ οι κρίσιμες διεργασίες δεν στερούνται πόρων, πάντα ανταποκρίνονται στις προθεσμίες τους και μία διεργασία δεν επιτρέπεται να λιμοκτονήσει.



Σχήμα 4.26 Προσαρμοστική τμηματοποίηση.

Η προσαρμοστική τμηματοποίηση του QNX προσφέρει στις εφαρμογές την αχρησιμοποίητη παραγωγική ικανότητα της CPU και ταυτόχρονα την παροχή εγγυήσεων που απαιτούνται σε κύκλους CPU για τις κρίσιμες διαδικασίες.

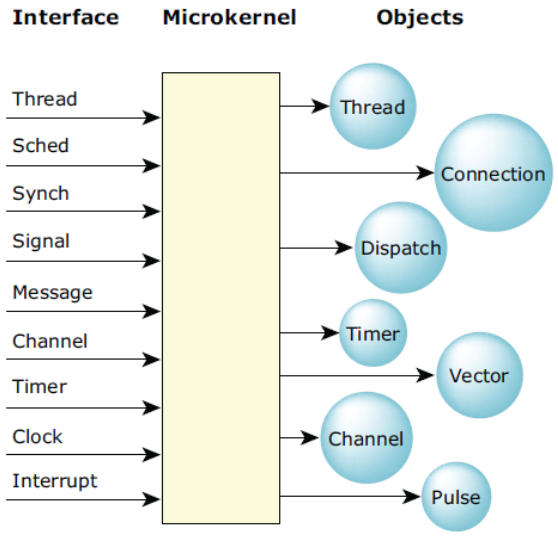
Ο σχεδιαστής του συστήματος μπορεί να διατηρήσει ένα ποσοστό των πόρων για ένα συγκεκριμένο τμήμα. Υπό κανονικές συνθήκες λειτουργίας, τα τμήματα μπορούν να χρησιμοποιήσουν όσους κύκλους CPU είναι διαθέσιμοι. Ωστόσο, κατά τη διάρκεια μιας υπερφόρτωσης, δηλαδή όταν απαιτείται περισσότερη εργασία από αυτή που το σύστημα μπορεί να αντέξει, η προσαρμοστική τμηματοποίηση επιβάλλει όρια για στους κύκλους CPU που κάθε τμήμα επιτρέπεται να καταναλώνει, έτσι εγγυάται ότι οι ελάχιστοι απαιτούμενοι πόροι της CPU είναι πάντα διαθέσιμες για συγκεκριμένες διεργασίες.

Ένα άλλο σημαντικό στοιχείο του είναι η γρήγορη εκκίνηση. Το QNX Neutrino προσφέρει αρκετές στρατηγικές για να πετύχει την εκκίνηση στους ελάχιστους χρόνους που απαιτούν πολλά ενσωματωμένα συστήματα. Μερικές είναι οι BIOS-less boot, microkernel και Instant Device Activation (IDA).

Τέλος ένα ενδιαφέρον χαρακτηριστικό του είναι ότι ενώ είναι ένα εμπορικό προϊόν παρέχεται τελείως δωρεάν για ακαδημαϊκή χρήση και μη εμπορική χρήση με μία απλή εγγραφή στην ιστοθέση του.

The QNX Neutrino Microkernel [33]

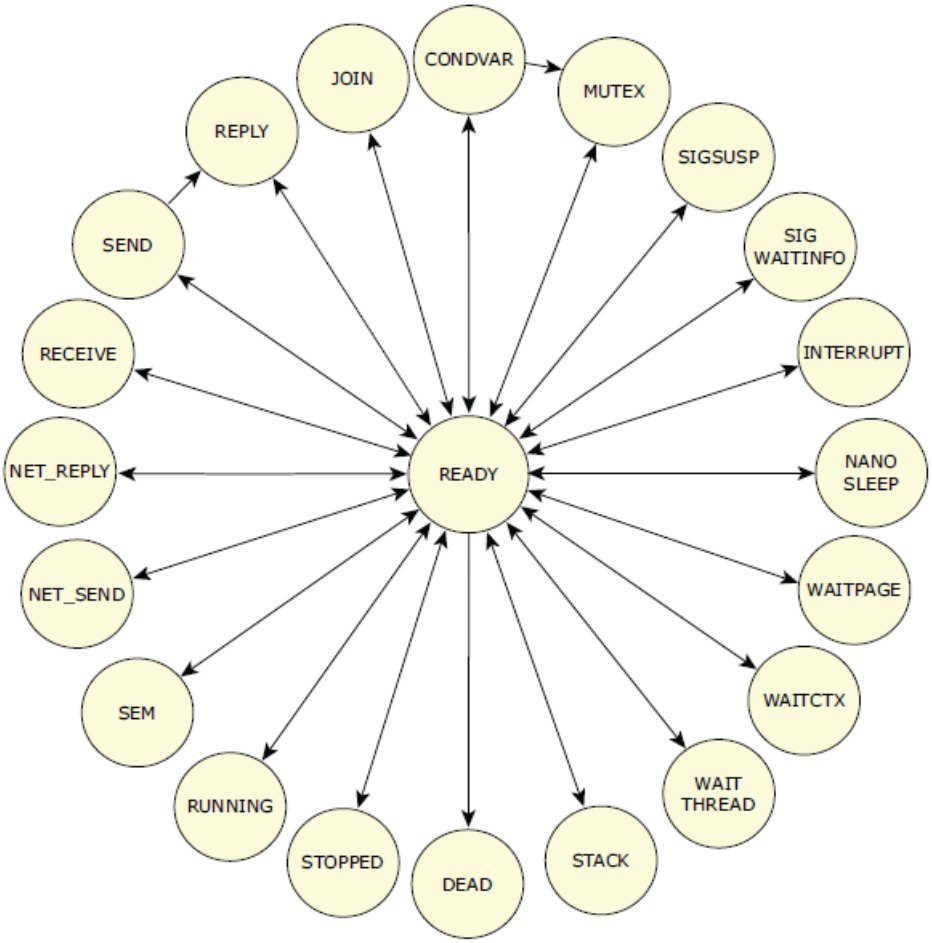
Ο μικροπυρήνας του QNX Neutrino (procnto), υλοποιεί τις βασικές λειτουργίες POSIX που χρησιμοποιούνται σε ενσωματωμένα συστήματα πραγματικού χρόνου, μαζί με τις θεμελιώδεις υπηρεσίες διαβίβασης μηνύματος του QNX Neutrino. Τα χαρακτηριστικά POSIX που δεν εφαρμόζονται στον μικροπυρήνα (σύστημα αρχείων και συσκευές εισόδου εξόδου, για παράδειγμα) παρέχονται σαν προαιρετικές διεργασίες και μοιραζόμενες βιβλιοθήκες. Στο χαμηλότερο επίπεδο του περιέχει μερικά θεμελιώδη αντικείμενα και τις ρουτίνες που τα χειρίζονται. Στο σχήμα 4.27 απεικονίζεται αυτή η δομή.



Σχήμα 4.27 Ο μικροπυρήνας του QNX.

Ο μικροπυρήνας του QNX Neutrino έχει κλήσεις πυρήνα για να υποστηρίξει: threads, message passing, signals, clocks, timers, interrupt handlers, semaphores, mutual exclusion locks (mutexes), condition variables (condvars) και barriers. Όλο το λειτουργικό σύστημα είναι χτισμένο πάνω σε αυτές τις κλήσεις και είναι πλήρως προεκτοπιστικό.

Νήματα και διεργασίες.



Σχήμα 4.28 Οι καταστάσεις ενός νήματος QNX.

Το νήμα μπορεί να θεωρηθεί σαν η ελάχιστη εκτελέσιμη μονάδα. Η διεργασία μπορεί να θεωρηθεί σαν ένας χώρος διευθύνσεων όπου εκτελούνται νήματα. Μία διεργασία αποτελείται κατ' ελάχιστο από ένα νήμα.

Το λειτουργικό σύστημα μπορεί να ρυθμιστεί ώστε να παρέχει ένα συνδυασμό νημάτων και διεργασιών (όπως ορίζονται κατά POSIX). Κάθε διεργασία είναι MMU-προστατευμένη από κάθε άλλη. Κάθε διεργασία περιέχει ένα ή περισσότερα νήματα που μοιράζονται το χώρο διευθύνσεων της διεργασία τους. Το QNX επιτρέπει την ύπαρξη 4095 διεργασιών με 32767 νήματα ανά διεργασία [33]. Οι διεργασίες δεν έχουν προτεραιότητες. Προτεραιότητες έχουν μόνο τα νήματα. Ο αριθμός των νημάτων μίας διεργασίας μπορεί να κυμανθεί σε μεγάλο εύρος γιατί τα νήματα δημιουργούνται και καταργούνται δυναμικά.

Τα βασικά χαρακτηριστικά ενός νήματος περιλαμβάνουν tid, προτεραιότητα, όνομα, σύνολο καταχωρητών, στοίβα, signal mask, δική του μνήμη, χειριστή τερματισμού. Οι βασικές καταστάσεις είναι ready και blocked. Στο σχήμα 4.28 φαίνονται όλες οι δυνατές καταστάσεις που μπορεί να βρεθεί ένα νήμα.

Χρονοπρογραμματισμός.

Η εκτέλεση ενός νήματος αναστέλλεται προσωρινά κάθε φορά που ο μικροπυρήνας δέχεται μια κλήση πυρήνα, μία εξαίρεση, ή ένα σήμα διακοπής από το υλικό. Μία απόφαση χρονοπρογραμματισμού λαμβάνεται κάθε φορά που η κατάσταση εκτέλεσης ενός νήματος αλλάζει. Τα νήματα χρονοπρογραμματίζονται συνολικά σε όλες τις διεργασίες.

Κανονικά, η εκτέλεση ενός νήματος σε αναστολή θα συνεχιστεί, αλλά ο χρονοπρογραμματιστής θα πραγματοποιήσει μία μεταγωγή περιβάλλοντος από το ένα νήμα στο άλλο κάθε φορά που το νήμα που εκτελείται :

- μπλοκάρεται
- προεκτοπίζεται
- αποχωρεί

Σε κάθε νήμα έχει εκχωρηθεί μία προτεραιότητα. Ο χρονοπρογραμματιστής επιλέγει για να τρέξει το επόμενο νήμα που είναι έτοιμο και έχει την υψηλότερη προτεραιότητα. Το λειτουργικό σύστημα υποστηρίζει συνολικά 256 επίπεδα προτεραιότητας προγραμματισμού. Ένα νήμα που δεν είναι αρχή μπορεί να έχει προτεραιότητά από 1 έως 63 (η υψηλότερη προτεραιότητα), ανεξάρτητα από την πολιτική χρονοπρογραμματισμού. Μόνο τα νήματα που είναι ρίζες (δηλαδή εκείνων των οποίων ενεργό UID, είναι 0) επιτρέπεται να έχουν προτεραιότητα πάνω από το 63. Ένα ειδικό νήμα το idle έχει προτεραιότητα 0 και είναι πάντα έτοιμο να τρέξει. Ένα νήμα εξ ορισμού κληρονομεί την προτεραιότητα του γονικού νήματος. Τα νήματα στην ουρά των ετοίμων ταξινομούνται κατά προτεραιότητα.

Αλγόριθμοι χρονοπρογραμματισμού.

Για την κάλυψη των αναγκών των διαφόρων εφαρμογών το QNX Neutrino παρέχει τους ακόλουθους αλγόριθμους χρονοπρογραμματισμού:

- FIFO.
- round-robin.
- Sporadic.

Ένα νήμα μπορεί να έχει όποια μέθοδο απαιτείται. Ο χρονοπρογραμματισμός γίνεται κατά νήμα. Οι μέθοδοι FIFO και Round Robin εφαρμόζονται σε νήματα που έχουν την ίδια προτεραιότητα.

Η προτεραιότητα ενός νήματος μπορεί να ποικίλλει κατά τη διάρκεια της εκτέλεσής του, είτε από αλλαγές από το ίδιο το νήμα ή από τον πυρήνα που προσαρμόζει την προτεραιότητα του νήματος καθώς λαμβάνει μήνυμα από ένα υψηλότερης προτεραιότητας νήμα.

Υπηρεσίες συγχρονισμού.

Οι υπηρεσίες συγχρονισμού περιλαμβάνουν:

Synchronization service Supported	**between processes**	**across a QNX LAN**
Mutexes	Yes	No
Condvars	Yes	No
Barriers	No	No
Sleepon locks	No	No
Reader/writer locks	Yes	No
Semaphores	Yes	Yes (named only)
FIFO scheduling	Yes	No
Send/Receive/Reply	Yes	Yes
Atomic operations	Yes	No

Οι παραπάνω υπηρεσίες υλοποιούνται άμεσα από τον πυρήνα, εκτός από Barriers, Sleepon locks και Reader/writer locks (τα οποία είναι κατασκευασμένα από mutexes και condvars) και ατομικές λειτουργίες (οι οποίες είτε υλοποιούνται άμεσα είτε προσομοιώνονται στον πυρήνα),

Mutexes, είναι η απλούστερη από τις υπηρεσίες συγχρονισμού. Χρησιμοποιείται για να εξασφαλίσει αποκλειστική πρόσβαση στα δεδομένα που μοιράζονται μεταξύ νημάτων. Μόνο ένα νήμα μπορεί να έχει κλειδώσει το mutex σε κάθε δεδομένη στιγμή. Νήματα που προσπαθούν να κλειδώσουν ήδη κλειδωμένα mutex θα μπλοκάρουν μέχρι το νήμα που κατέχει το mutex τον ξεκλειδώσει. Όταν το νήμα ξεκλειδώνει το mutex, το νήμα με την υψηλότερη προτεραιότητα από αυτά που περιμένουν για να κλειδώσουν το mutex θα ξεμπλοκάρει και να γίνει ο νέος ιδιοκτήτης του mutex.

Μια μεταβλητή συνθήκης ή condvar, χρησιμοποιείται για να μπλοκάρει ένα νήμα μέσα σε ένα κρίσιμο τμήμα μέχρι κάποια προϋπόθεση να ικανοποιείται. Η προϋπόθεση μπορεί να είναι οσοδήποτε σύνθετη και είναι ανεξάρτητα από το condvar. Ωστόσο, η condvar πρέπει να χρησιμοποιούνται πάντα με ένα mutex κλείδωμα για να υλοποιεί ένα έλεγχο.

Μια condvar υποστηρίζει τρεις λειτουργίες:

- αναμονής (pthread_cond_wait ())
- σήματος (pthread_cond_signal ())
- εκπομπής (pthread_cond_broadcast ())

Barrier είναι ένας μηχανισμός συγχρονισμού που επιτρέπει να «μαζευτούν» αρκετά συνεργαζόμενα νήματα και να αναγκαστούν να περιμένουν σε ένα συγκεκριμένο σημείο έως ότου όλα τα νήματα έχουν τελειώσει πριν το κάθε ένα μπορεί να συνεχιστεί.

Οι σηματοφορείς είναι μια άλλη μορφή συγχρονισμού. Μια σημαντική διαφορά μεταξύ σηματοφορέων και άλλων μεθόδων συγχρονισμού είναι ότι σηματοφόροι είναι «ασύγχρονα ασφαλείς" και μπορούν να χρησιμοποιηθούν από τους χειριστές σήματος. Εάν το ζητούμενο είναι να έχουμε ένα χειριστή σήματος να ξυπνήσει ένα νήμα, οι σηματοφόροι είναι η σωστή επιλογή.

Σε μονοεπεξεργαστικά συστήματα ο χρονοπρογραμματισμός κατά FIFO εγγυάται ότι δύο νήματα ίδιας προτεραιότητας δεν θα εκτελέσουν ταυτόχρονα τα κρίσιμα τμήματά τους.

Ο συγχρονισμός μέσω διαβίβασης μηνύματος είναι η μόνη μέθοδος που μπορεί να χρησιμοποιηθεί και σε ένα δίκτυο. Ο συγχρονισμός μέσω ατομικών λειτουργιών είναι: adding a value, subtracting a value, clearing bits, setting bits, toggling (complementing) bits

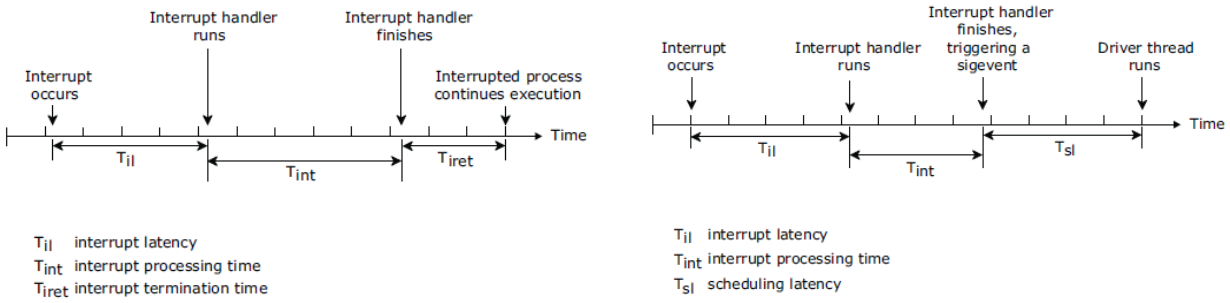
Χειρισμός διακοπών.

Στα συστήματα πραγματικού χρόνου ο χρόνος καθυστέρησης που είναι ο χρόνος που μεσολαβεί από την έλευση ενός αιτήματος εξυπηρέτησης μέχρι τη στιγμή που εκκινεί η εκτέλεση του κώδικα που θα εξυπηρετήσει αυτή την αίτηση είναι ένα σημαντικό χαρακτηριστικό του συστήματος.

Οι δύο χρονικές καθυστερήσεις που είναι σημαντικές είναι η καθυστέρηση εξυπηρέτησης των σημάτων διακοπής και η καθυστέρηση χρονοπρογραμματισμού. Αυτοί οι χρόνοι είναι στενά εξαρτώμενοι και από την τεχνολογία του υποκείμενου υλικού.

Ο χρόνος καθυστέρησης εξυπηρέτησης σημάτων διακοπής είναι πολύ μικρός αλλά ορισμένα κρίσιμα τμήματα κώδικα απαιτούν προσωρινά να απενεργοποιηθεί ο μηχανισμός υποδοχής σημάτων διακοπής. Ο μέγιστη χρόνος απενεργοποίησης καθορίζει συνήθως την χειρότερη καθυστέρηση.

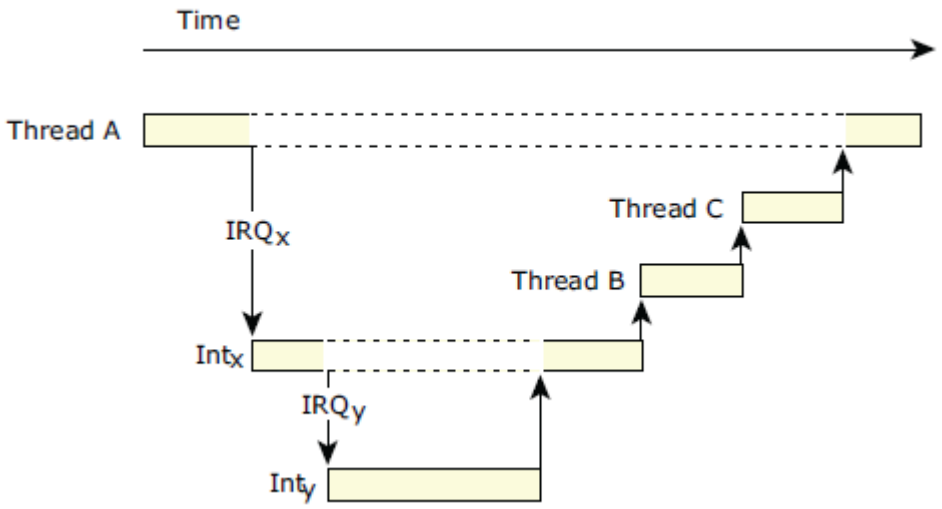
Τα παρακάτω διαγράμματα απεικονίζουν την περίπτωση όπου μια διακοπή υλικού υποβάλλεται σε επεξεργασία από ένα χειριστή διακοπής. Ο χειριστής είτε απλά θα επιστρέψει είτε θα επιστρέψει και θα προκαλέσει ένα γεγονός που πρέπει να παραδοθεί.



Σχήμα 4.29 Λανθάνον χρόνος διακοπής και χρονοπρογραμματισμού.

Σε ορισμένες περιπτώσεις, ο χαμηλού επιπέδου χειριστής διακοπής υλικού πρέπει να προγραμματίσει ένα υψηλότερου επιπέδου νήμα για να τρέξει. Σε αυτό το σενάριο, ο χειριστής διακοπής θα επιστρέψει και ένα συμβάν πρέπει να παραδοθεί. Αυτό εισάγει μια δεύτερη μορφή της καθυστέρησης, την καθυστέρηση χρονοπρογραμματισμού. Αυτός ο χρόνος είναι ο χρόνος που χρειάζεται για να αποθηκευτεί το περιβάλλον εργασίας του τρέχοντος νήματος και να αποκατασταθεί το περιβάλλον εργασίας του οδηγούμενου νήματος. Αν και μεγαλύτερη από την καθυστέρηση διακοπών σε ένα σύστημα QNX Neutrino είναι μικρός.

Το QNX Neutrino υποστηρίζει πλήρως την διακοπή ενός σήματος διακοπής από ένα άλλο υψηλότερης προτεραιότητας.



Σχήμα 4.30 Stacked IRQs.

Διαδιεργασιακή επικοινωνία (IPC).

Η διαδιεργασική επικοινωνία διαδραματίζει θεμελιώδη ρόλο στη μεταμόρφωση του QNX Neutrino από ένα ενσωματωμένο πυρήνα πραγματικού χρόνου σε ένα λειτουργικό σύστημα συμβατό με POSIX. Δεδομένου ότι οι διάφορες διεργασίες που παρέχουν υπηρεσίες προστίθεται στο μικροπυρήνα η IPC είναι η κόλλα που συνδέει αυτά τα στοιχεία σε ένα συνεκτικό σύνολο.

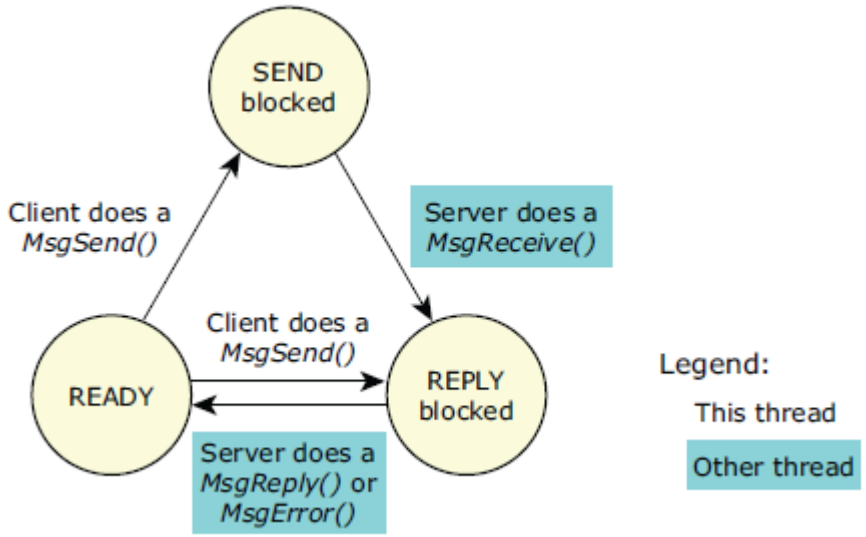
Αν και η διαβίβαση μηνύματος είναι η πρωταρχική μορφή της IPC υπάρχουν και άλλοι τύποι. Αυτοί οι τύποι είναι χτισμένοι πάνω από τη βασική που είναι η διαβίβαση μηνύματος. Η στρατηγική είναι να δημιουργήσει μία απλή και εύρωστη υπηρεσία IPC που μπορεί να ρυθμιστεί για απόδοση μέσω ενός απλοποιημένου κώδικα στο μικροπυρήνα και μετά νέες υπηρεσίες να μπορούν να υλοποιηθούν πάνω από τη βασική.

Το QNX Neutrino έχει τις ακόλουθες μορφές IPC:

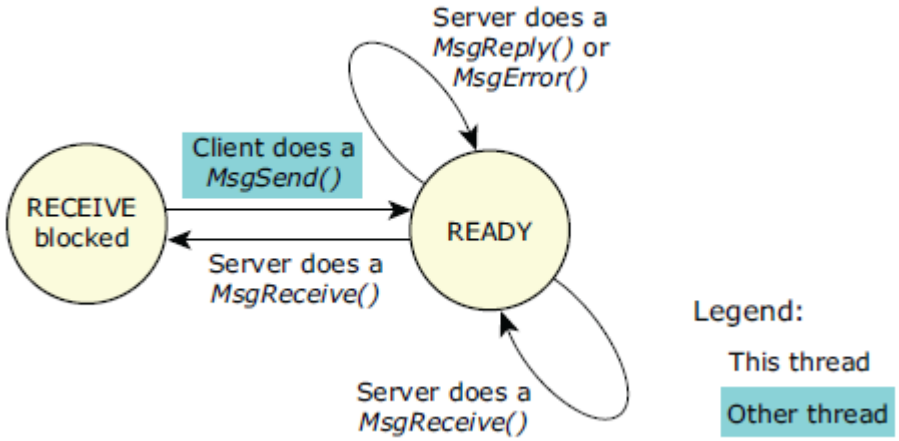
Service:	**Implemented in**
Message-passing	Kernel
Signals	Kernel
POSIX message queues	External process
Shared memory	Process manager
Pipes	External process
FIFOs	External process

Σύγχρονη διαβίβαση μηνύματος. Ένα νήμα που κάνει MsgSend () σε ένα άλλο νήμα θα είναι δεσμευμένο μέχρι το νήμα στόχος κάνει MsgReceive (), να επεξεργαστεί το μήνυμα και να εκτελέσει μια MsgReply (). Αν ένα νήμα εκτελεί μια MsgReceive () χωρίς προηγουμένως να σταλεί μήνυμα, θα μπλοκάρει μέχρι ένα άλλο νήμα να εκτελέσει μια MsgSend ().

Σε Neutrino, ένα νήμα διακομιστής συνήθως επαναλαμβάνεται, περιμένοντας να λάβει ένα μήνυμα από έναν νήμα πελάτη. Όπως περιγράφεται παραπάνω, ένα νήμα είναι στην κατάσταση READY εάν πρόκειται να χρησιμοποιήσετε τη CPU. Δεν μπορεί στην πραγματικότητα να πάρει οποιοδήποτε χρόνο CPU λόγω της προτεραιότητας και του αλγόριθμο χρονοπρογραμματισμού αλλά το νήμα δεν είναι μπλοκαρισμένο.



Σχήμα 4.31 Το νήμα πελάτη.



Σχήμα 4.32 Το νήμα εξυπηρετητή.

Το κληρονομούμενο μπλοκάρισμα συγχρονίζει την εκτέλεση του νήματος αποστολέα, αφού η πράξη να ζητούν ότι τα δεδομένα θα σταλούν, επίσης, προκαλεί το μπλοκάρισμα του νήματος αποστολέα και το νήμα παραλήπτης χρονοπρογραμματίζονται για εκτέλεση. Αυτό συμβαίνει χωρίς να απαιτείται ρητή εργασία από τον πυρήνα που να καθορίσει ποια νήμα να διεξαχθεί ως επόμενο (όπως θα συνέβαινε με τις περισσότερες άλλες μορφές IPC). Εκτέλεση και δεδομένα μετακινούνται απευθείας από το ένα πλαίσιο στο άλλο. Το QNX υποστηρίζει τα 32 τυπικά σήματα του POSIX αλλά και τα POSIX σήματα πραγματικού χρόνου.

Πλατφόρμες υλικού.

Το QNX υποστηρίζει τους επεξεργαστές

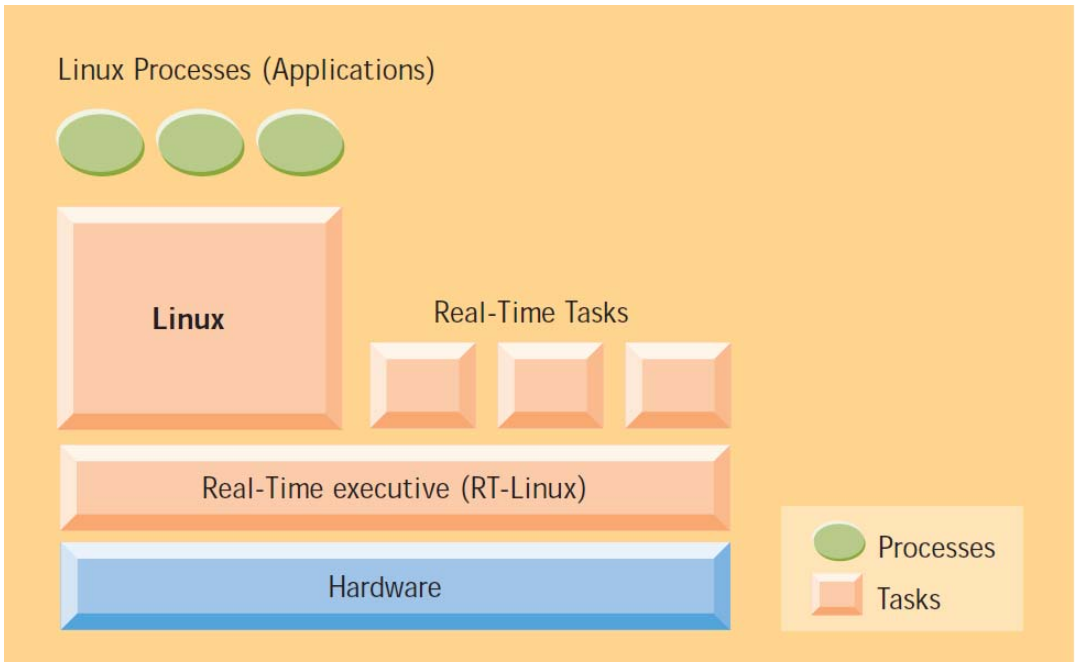
- της οικογένειας x86
- της οικογένειας ARM
- της οικογένειας XScale
- της οικογένειας PowerPC
- της οικογένειας MIPS
- της οικογένειας SH-4

4.3 Λειτουργικά συστήματα πραγματικού χρόνου ανοικτού λογισμικού.

4.3.1 Linux.

Στο ανοικτό λογισμικό μία λύση με πάρα πολλές παραλλαγές είναι οι διάφορες τροποποιημένες εκδόσεις του Linux. Επειδή οι παραλλαγές του είναι πολλές και κάθε μία έχει τις δικές της βελτιώσεις, εδώ θα περιγράψουμε τον γενικό τρόπο με το οποίο το Linux από ένα γενικό λειτουργικό σύστημα μετατρέπεται σε ένα λειτουργικό σύστημα πραγματικού χρόνου.

Το Linux δεν μπορεί να χειριστεί εργασίες πραγματικού χρόνου. Μπορεί να ανταποκριθεί σε γεγονότα μέσα σε ένα προκαθορισμένο χρονικό διάστημα κατά μέσο όρο, αλλά αυτό δεν είναι πάντα αποδεκτό. Υπάρχουν αρκετοί λόγοι για αυτό, αλλά ο κυριότερος είναι ότι ορισμένες λειτουργίες του πυρήνα δεν μπορεί να είναι προεκτοπιστικές. Ως εκ τούτου, όταν ακόμα και μία χαμηλής προτεραιότητας διαδικασία εκτελεί αυτό το είδος της λειτουργίας, δεν μπορεί να διακοπεί υπέρ μιας διαδικασίας με υψηλότερη προτεραιότητα. Αυτό δεν είναι αποδεκτό, διότι η διαδικασία υψηλής προτεραιότητας θα πρέπει σε όλες τις περιπτώσεις, να ενεργοποιηθεί άμεσα. Για να επιτευχθεί το απαιτούμενο επίπεδο ντετερμινισμού, πρέπει να προστεθούν στο Linux κάποιες επεκτάσεις. Δύο από τις διαθέσιμες προτάσεις είναι το RTAI (Real Time Application Interface) και το Real-Time Linux (RTLinux) που βασίζονται στην ίδια αρχή. Σε αντίθεση με ένα πλήρες λειτουργικό σύστημα πραγματικού χρόνου εδώ έχουμε εκτέλεση πραγματικού χρόνου στην οποία η χαμηλότερης προτεραιότητας εργασία (με την έννοια της εκτέλεσης) είναι το ίδιο το Linux στο σύνολό του όπως φαίνεται στο σχήμα 4.33.



Σχήμα 4.33 Η δομή του RT-Linux [28].

Το RTLinux ή το RTAI παίρνει τον έλεγχο της μηχανής, και παρακρατά όλες τις διακοπές που κανονικά απευθύνεται στο Linux και τις αναδιαθέτει σε αυτό μόνο όταν όλες οι πραγματικού χρόνου εργασίες έχουν ολοκληρωθεί. Η επικοινωνία μεταξύ των διεργασιών του Linux γίνεται με διάφορους πόρους επικοινωνίας. Με αυτό τον τρόπο, οι εργασίες που απαιτούν λειτουργικότητα σε πραγματικό χρόνο μπορούν να συμβιώνουν με άλλες λιγότερο απαιτητικές εφαρμογές στο ίδιο μηχάνημα. Οι εφαρμογές πραγματικού χρόνου που αναπτύσσονται για το RTLinux και το RTAI είναι συμβατές με το πρότυπο POSIX. Σήμερα η πιο συχνά χρησιμοποιούμενη επέκταση είναι το RTLinux [34].

Το RTLinux αναπτύχθηκε αρχικά στο Ινστιτούτο Τεχνολογίας του Νέου Μεξικού. Αλλά αργότερα χωρίστηκε σε δύο τμήματα: στο Open RTLinux που είναι ελεύθερο και προορίζεται για την έρευνα και την ακαδημαϊκή κοινότητα και το Wind River RT Core που αναπτύσσεται από την Wind River για εμπορικούς σκοπούς και έχει επαγγελματική υποστήριξη.

4.3.2 eCos - embedded Configurable Operating System (www.ecos.sourceware.org)

Οι βασικές σχεδιαστικές αρχές του eCOS ήταν να έχει μικρές απαιτήσεις σε μνήμη, μικρές απαιτήσεις σε πόρους, να είναι οικονομικό και να είναι λύση υψηλής ποιότητας για την αγορά του ενσωματωμένου λογισμικού [36].

Το eCOS είναι χωρίς χρεώσεις και δικαιώματα. Οι προγραμματιστές έχουν πλήρη πρόσβαση σε ολόκληρο τον πηγαίο κώδικα, συμπεριλαμβανομένων των εργαλείων, τα οποία μπορούν να τροποποιηθούν ανάλογα με τις ανάγκες τους. Η ιδιαίτερα διαμορφώσιμη φύση του έδωσε τη δυνατότητα στις εταιρίες που το χρησιμοποιούν, να μειώσουν το χρόνου διάθεσης στην αγορά για τα προϊόντα τους.

Το ενσωματωμένο λογισμικό σήμερα παρέχει περισσότερες λειτουργίες από ότι μπορεί στην πραγματικότητα να απαιτείται από μια συγκεκριμένη εφαρμογή. Συχνά, επιπλέον κώδικας περιλαμβάνεται σε ένα σύστημα και δίνει υποστήριξη για λειτουργίες που δεν χρειάζονται για την συγκεκριμένη εφαρμογή. Αυτός ο επιπλέον κώδικας κάνει το λογισμικό πιο περίπλοκο χωρίς λόγο. Επιπλέον, όσο μεγαλύτερος ο κώδικας, τόσο μεγαλύτερη είναι η πιθανότητα να πάει κάτι στραβά.

Στο eCOS οι προγραμματιστές έχουν τη δυνατότητα να επιλέξουν τα στοιχεία που ικανοποιούν τις βασικές ανάγκες της εφαρμογής και να τα ρυθμίσουν για τις ειδικές απαιτήσεις και τις λεπτομέρειες της εφαρμογής. Οι δυνατότητες αύξησης της λειτουργικότητας του eCOS είναι απεριόριστες. Εάν η λειτουργικότητα που απαιτείται δεν είναι διαθέσιμη ο πηγαίος κώδικας είναι ανοικτός για να υλοποιήσει κάποιος μόνος του την εργασία που θέλει.

Το eCos είναι σήμερα το πιο αποδεκτό RTOS ανοιχτού κώδικα. Το περιβάλλον ανάπτυξης είναι διαθέσιμο για windows και για Linux. Προορίζεται κύρια για ενσωματωμένα συστήματα και εφαρμογές που απαιτούν μόνο μία διεργασία με πολλαπλά νήματα, και έχουν περιορισμένη μικρή μνήμη της τάξης των μερικών εκατοντάδων Kbytes

Δεδομένου ότι τα ενσωματωμένα συστήματα γίνονται μικρότερα, ταχύτερα, φθηνότερα, και πιο εξελιγμένα, ο έλεγχος πάνω σε όλο το λογισμικό του συστήματος είναι απαραίτητος. Η φιλοσοφία του eCOS είναι να μειωθεί το μέγεθος για τα συστήματα που έχουν περιορισμένους πόρους, ακόμη και εις βάρος των συστημάτων που δεν έχουν περιορισμούς πόρων. Λόγω αυτής της σχεδιαστικής φιλοσοφία σχεδιασμού τα συστήματα δεν πάσχουν από πρόσθετο κώδικα αναγκαίο μόνο στην υποστήριξη προηγμένων σύνθετων συστημάτων.

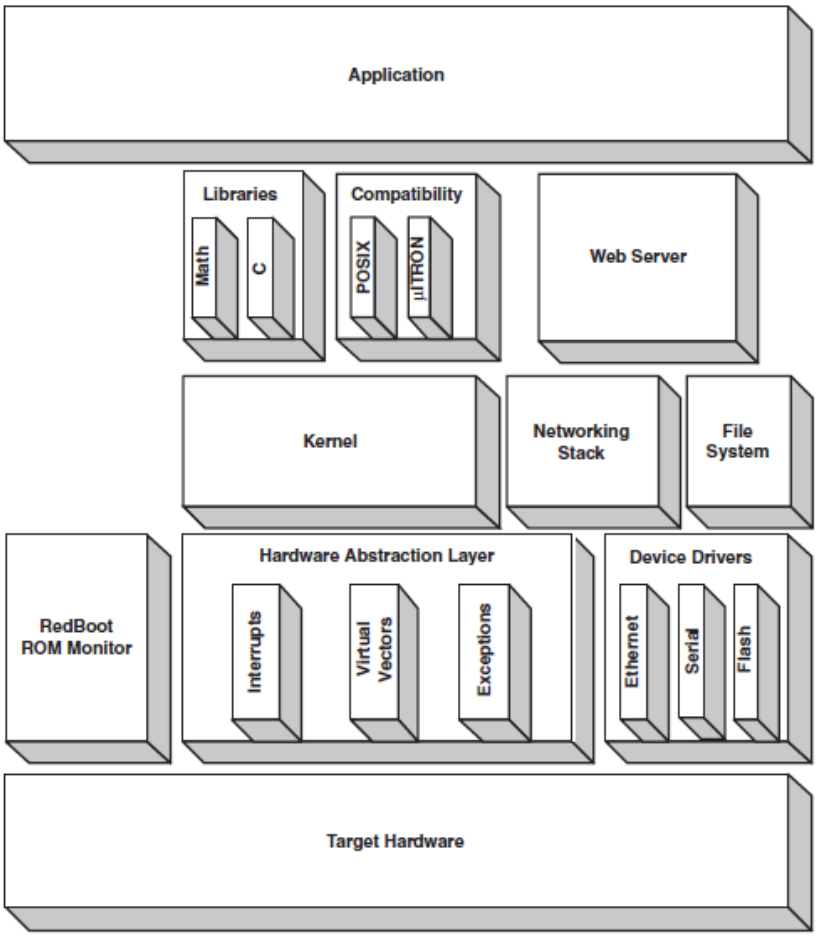
Τα βασικά στοιχεία του είναι τα εξής[37]:

- Hardware Abstraction Layer (HAL) - παρέχει ένα στρώμα λογισμικού που δίνει πρόσβαση στο υλικό.
- Kernel - συμπεριλαμβάνει το χειρισμό εξαιρέσεων, χειρισμό σημάτων διακοπής, τα νήματα και το συγχρονισμό τους, τις υλοποιήσεις χρονοπρογραμματισμού, χρονοδιακόπτες, μετρητές, και συναγερμούς.
- ISO C and math libraries.
- οδηγοί συσκευών - συμπεριλαμβάνει τη βασική σειριακή επικοινωνία, Ethernet, Flash ROM, και άλλα.

- GNU debugger (GDB) support - χρήση GDB για την εκσφαλμάτωση της εφαρμογής.

Στο eCOS δεν υπάρχει διάκριση μεταξύ κατάστασης χρήστη και κατάσταση πυρήνα. Το eCOS και η εφαρμογή εκτελούνται σε κατάσταση επόπτη. Περιλαμβάνει και μία υποδομή δοκιμών που βελτιώνεται συνεχώς στις νεώτερες εκδόσεις.

Η αρχιτεκτονική του έχει θεμελιώδη στόχο να επιτρέψει την κατασκευή ενός πλήρους ενσωματωμένου συστήματος από επαναχρησιμοποιήσιμα δομικά στοιχεία λογισμικού. Το σχήμα 4.34 παρουσιάζεται ένα παράδειγμα του πώς τα βασικά δομικά στοιχεία του πυρήνα και ορισμένα από τα προαιρετικά στοιχεία μπορούν να ενωθούν για τις απαιτήσεις μιας πολύπλοκης εφαρμογής.



Σχήμα 4.34 Η δομή μίας σύνθετης εφαρμογής στο eCOS.

Hardware Abstraction Layer (HAL)

Το Hardware Abstraction Layer (HAL) είναι το στοιχείο λογισμικού που θα πρέπει να τροποποιηθεί κατάλληλα όταν το eCOS πρέπει να μεταφερθεί σε ένα συγκεκριμένο υλικό. Το HAL απομονώνει τα χαρακτηριστικά που εξαρτώνται από την αρχιτεκτονική του υλικού και τα παρουσιάζει σε μια γενική μορφή που επιτρέπει τη φορητότητα των άλλων συνιστωσών της υποδομής. Το HAL είναι ένα στρώμα του λογισμικού, με ένα γενικό Application Programming Interfaces (API), που καλύπτει τις ειδικές λειτουργίες του υλικού για να ολοκληρωθεί μια επιθυμητή εργασία.

Διακοπές και εξαιρέσεις.

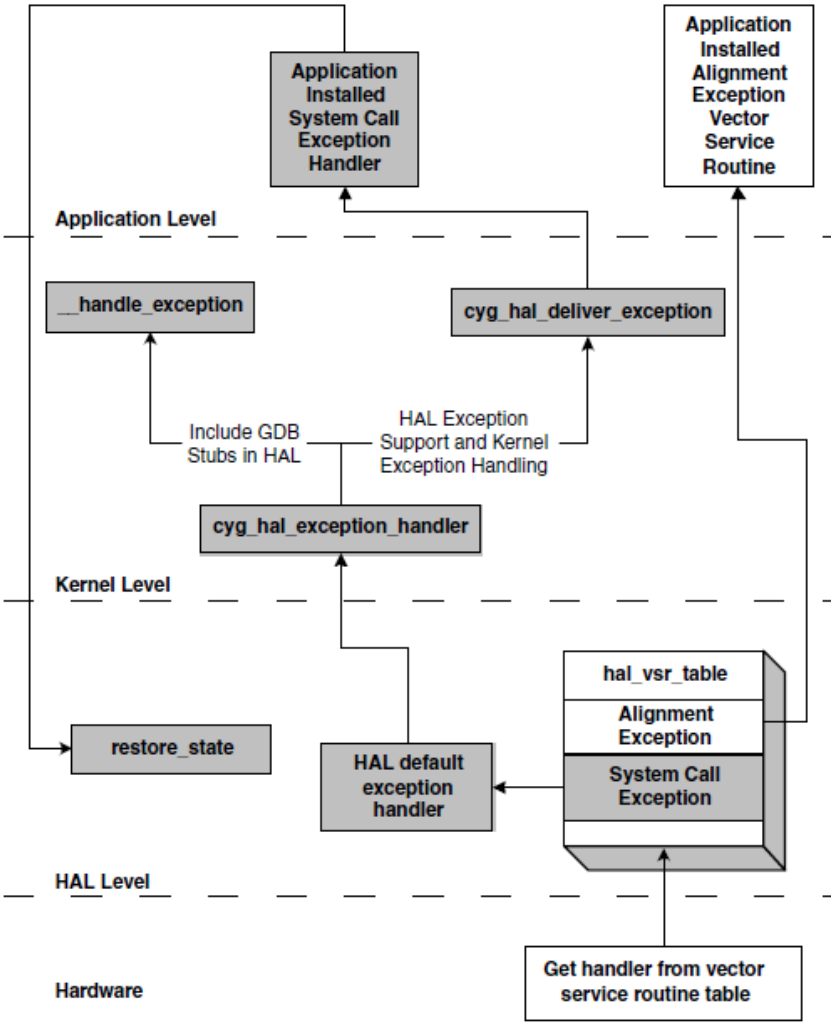
Μια εξαίρεση είναι ένα σύγχρονο γεγονός που συνέβη κατά την εκτέλεση ενός νήματος και διαταράσσει την κανονική ροή των εντολών του. Αν οι εξαιρέσεις δεν έχουν σωστή μεταχείριση μπορεί οι συνέπειες για το σύστημα να είναι σοβαρές. Ο χειρισμός των εξαιρέσεων είναι εξαιρετικά σημαντικός παράγοντας για την εύρυθμη λειτουργία του λο-

γισμικού που μπορεί να βοηθήσει και στην ανάκαμψη του λογισμικού μετά από μια εξαίρεση που συνέβη.

Εξαιρέσεις μπορούν να συμβούν σε ένα σύστημα από το υλικό (για παράδειγμα ένα σφάλμα πρόσβασης μνήμης) και από το λογισμικό (για παράδειγμα διαίρεση με μηδέν).

Η απλούστερη και πιο ευέλικτη μέθοδος για το χειρισμό των εξαιρέσεων είναι να κληθεί μια συνάρτηση. Αυτή η συνάρτηση χρειάζεται ένα πλαίσιο ή μια περιοχή για να κάνει τη δουλειά της. Μετά την ολοκλήρωση της ρουτίνας εξυπηρέτησης της εξαίρεσης τον νήμα συνεχίζει την εκτέλεσή του.

Στο eCos υπάρχουν δύο μέθοδοι. Η πρώτη, που είναι και η εξ' ορισμού μέθοδος, συνδυάζει το HAL και το Kernel Exception Handling. Η δεύτερη είναι η Application Exception Handling. Αυτή επιτρέπει στην εφαρμογή να αναλάβει τον πλήρη έλεγχο κάποιας ή όλων των εξαιρέσεων και να αντιστοιχίσει απευθείας στο υλικό ένα διάνυσμα ρουτινών εξυπηρέτησης. Όταν χρησιμοποιηθεί αυτή τη μέθοδος η ρουτίνα χειρισμού εξαιρέσεων πρέπει να είναι γραμμένη σε assembly. Σχηματικά οι δύο μέθοδοι φαίνονται στο σχήμα 4.35.



Σχήμα 4.35 Εξυπηρέτηση μίας εξαίρεσης. Με γκρι η HAL και με λευκό η Application Exception Handling.

Τα σήματα διακοπής είναι ασύγχρονα εξωτερικά γεγονότα που συμβαίνει κατά τη διάρκεια εκτέλεσης του προγράμματος προκαλώντας διακοπή της κανονικής του εκτέλεσης. Συνήθως, αυτά τα εξωτερικά γεγονότα αφορούν κυρίως το υλικό, όπως το πάτημα ενός κουμπιού ή η λήξη ενός χρονομετρητή. Η διακοπή μπορεί να συμβεί οποιαδήποτε χρονική στιγμή. Οι διακοπές επιτρέπουν σε κρίσιμες χρονικά λειτουργίες να εκτελούνται με

μεγαλύτερη προτεραιότητα σε σχέση με την κανονική εκτέλεση του προγράμματος. Παρόμοια με το χειρισμό των εξαιρέσεων, όταν έρθει ένα σήμα διακοπής, ο επεξεργαστής κάνει άλμα σε μία συγκεκριμένη διεύθυνση για την εκτέλεση της ρουτίνας εξυπηρέτησης της διακοπής (ISR). Η υποστήριξη του υλικού για διακοπές ποικίλλει πολύ ανάμεσα στις διαφορετικές αρχιτεκτονικές. Κάθε επεξεργαστής έχει το δικό του αριθμό ακροδεκτών για την ενεργοποίηση του ISR. Οι μέθοδοι για τον χειρισμό του διανύσματος των διακοπών ανάμεσα σε διαφορετικές αρχιτεκτονικές είναι επίσης διαφορετικές. Μερικές αρχιτεκτονικές υποστηρίζουν την διανυσματοποίηση των διακοπών σε μεμονωμένα διανύσματα, ενώ άλλες έχουν ένα ενιαίο διάνυσμα για όλες τις διακοπές. Όταν υποστηρίζονται ατομική διανύσματα, μια ρουτίνα εξυπηρέτησης μπορεί να συνδεθεί άμεσα με το διάνυσμα για την εξυπηρέτηση της διακοπής. Για υποστήριξη ενός διανύσματος, το λογισμικό πρέπει να καθορίσει ποια διακοπή συνέβη πριν προβεί στην ενεργοποίηση της κατάλληλης ISR. Μία από τις βασικές ανησυχίες στα ενσωματωμένα συστήματα σε σχέση με τις διακοπές είναι ο χρόνος καθυστέρησης εξυπηρέτησης (Latency) που είναι το χρονικό διάστημα από το σημείο που εμφανίζεται το σήμα μέχρι να αρχίσει η εκτέλεση της ρουτίνας εξυπηρέτησης του.

Το eCOS για να μειώσει τον χρόνο καθυστέρησης στο σύστημα, χωρίζει την εξυπηρέτηση μίας διακοπής σε δύο μέρη. Το πρώτο μέρος είναι η ρουτίνα εξυπηρέτησης ISR και το δεύτερο μέρος είναι η Deferred Service Routine (DSR). Αυτό το σχήμα επιτρέπει την ελαχιστοποίηση της καθυστέρηση, μειώνοντας τον χρόνο που παραμένει το σύστημα μέσα στην ρουτίνα εξυπηρέτησης. Η ιδέα είναι να κρατήσει την επεξεργασία μέσα στην ISR στο ελάχιστο. Σε αυτό το σχήμα η DSR εκτελείται με ενεργοποιημένα τα σήματα διακοπής, επιτρέποντας σε άλλα υψηλότερης προτεραιότητας σήματα να συμβούν και να υφίστανται επεξεργασία στη μέση της εξυπηρέτησης διακοπής χαμηλότερης προτεραιότητας.

Σε ορισμένες περιπτώσεις, όπου υπάρχουν μικρές απαιτήσεις για να εξυπηρετηθεί, η διακοπή μπορεί να αντιμετωπιστεί μόνο με την ISR. Εάν η εξυπηρέτηση είναι πιο πολύπλοκη, θα πρέπει να χρησιμοποιηθεί και μία DSR. Η DSR εκτελείται σε μεταγενέστερο χρόνο, όταν επιτρέπεται ο χρονοπρογραμματισμός νημάτων. Εκτελώντας το DSR αργότερα, επιτρέπεται να χρησιμοποιήσει, στην DSR, τους μηχανισμούς συγχρονισμού του πυρήνα, όπως για παράδειγμα, να στείλει ένα σήμα σε ένα νήμα, μέσω ενός σηματοφορέα, ότι μια διακοπή έχει συμβεί. Ωστόσο, υπάρχουν περίοδοι κατά τις οποίες ο χρονοπρογραμματισμός νημάτων είναι απενεργοποιημένη από τον πυρήνα, αν και αυτές οι περίοδοι διατηρούνται όσο το δυνατόν μικρότερες.

Τα νήματα του χρήστη μπορούν να αναστείλουν τον χρονοπρογραμματισμό και αυτό εμποδίζει την DSR από τα να εκτελεστεί. Εμποδίζοντας την έγκαιρη εκτέλεση της DSR το σύστημα μπορεί να οδηγηθεί σε αστοχίες από μια υπερχρήση της πηγής των διακοπών. Τα θέματα αυτά πρέπει να λαμβάνονται υπόψη κατά το σχεδιασμό του συστήματος της δομής των διακοπών και της αλληλεπίδρασης του με τα νήματα στο σύστημα.

Στις περισσότερες περιπτώσεις, η DSR εκτελείται αμέσως μετά την ολοκλήρωση της ISR. Ωστόσο, εάν ένα νήμα έχει κλειδώσει το χρονοπρογραμματιστή, η DSR θα καθυστερήσει μέχρι το νήμα να τον ξεκλειδώσει. Το καθεστώς προτεραιότητας ορίζει ότι οι ISR έχουν απόλυτη προτεραιότητα έναντι των DSRs, και οι DSRs έχουν απόλυτη προτεραιότητα σε σχέση με τα νήματα.

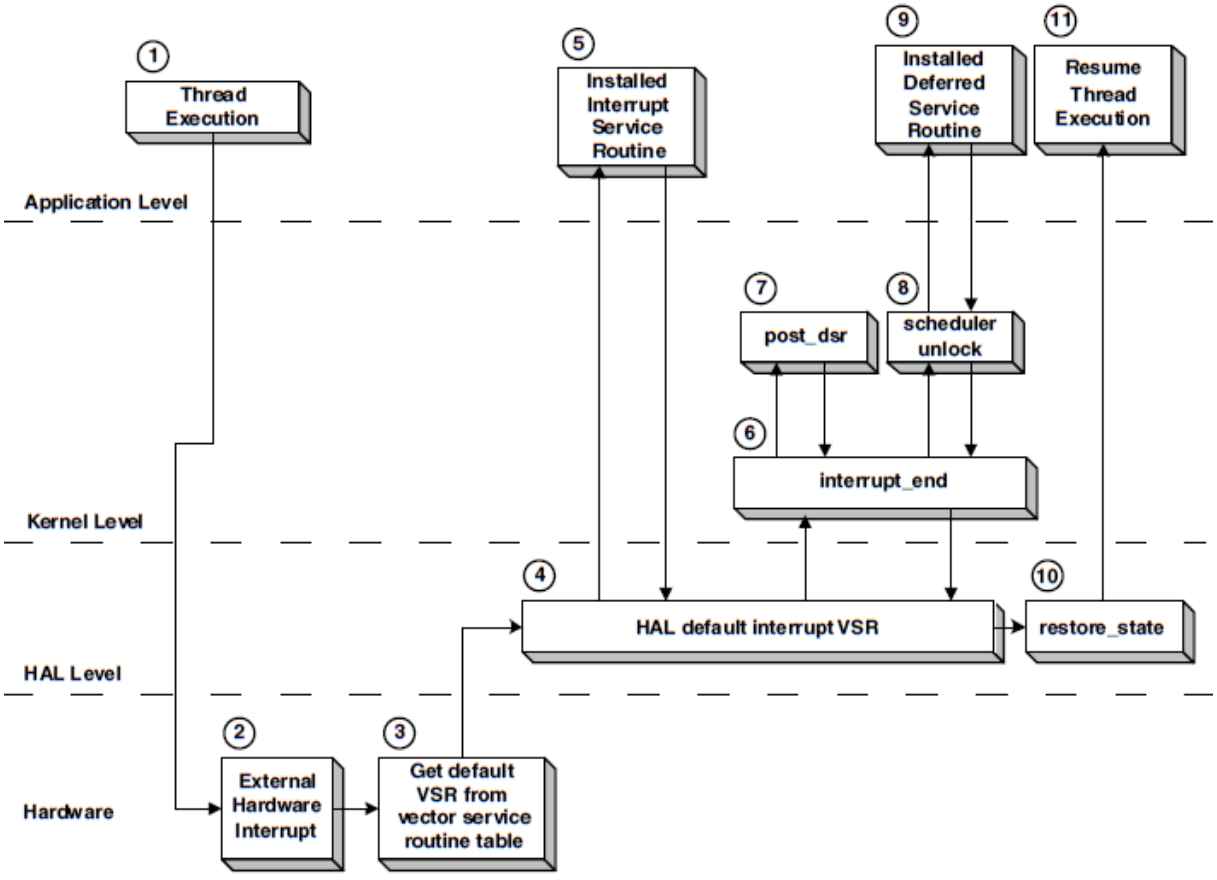
Για να λειτουργήσει σωστά αυτό το σχήμα πρέπει να διασφαλίζεται ότι η διακοπή που μόλις συνέβη δεν θα επαναληφθεί μέχρι το DSR να έχει ολοκληρώσει την επεξεργασία της. Για να επιτευχθεί αυτό, η ISR μασκάρει την τρέχουσα διακοπή και η DSR την ελευθερώνει αφού την έχει εξυπηρετήσει.

Μερικές υλοποιήσεις HAL προσφέρουν ένα σχήμα εμφωλευμένων διακοπών. Σε αυτή την περίπτωση, σήμα διακοπής μεγαλύτερης προτεραιότητα μπορεί να διακόψει και να εξυπηρετηθεί, πριν από το χαμηλότερης προτεραιότητας σήμα.

Για να εξασφαλίζεται η σωστή λειτουργία είναι απαραίτητο να ακολουθούνται ορισμένες αρχές κατά τη διάρκεια της επεξεργασίας ενός σήματος διακοπής. Κατ' αρχάς η ISR δεν μπορεί να κάνει οποιαδήποτε κλήση συνάρτησης που σχετίζεται με το συγχρονισμό του χρονοπρογραμματιστή. Αυτές περιλαμβάνουν συναρτήσεις του API του πυρήνα για σηματοφορείς, mutexes και μεταβλητές κατάστασης. Οι συναρτήσεις συγχρονισμού του πυρήνα προκαλούν αλληλεπίδραση με το χρονοπρογραμματιστή που είναι απενεργοποιημένος κατά την εκτέλεση της ISR. Κάνοντας αυτές τις κλήσεις προκαλείται απροσδιόριστη συμπεριφορά στο σύστημα που μπορεί να οδηγήσει σε κατάρρευσή του. Το σχήμα διακοπών του eCOS επιτρέπει αυτές οι κλήσεις για συγχρονισμό να γίνουν από την DSR.

Συνήθως, η DSR θα εκτελέσει αμέσως μετά την ολοκλήρωση της ISR. Δεδομένου ότι η DSR εκτελείται με ενεργοποιημένο τον χρονοπρογραμματιστή νημάτων, επιτρέπονται ορισμένες κλήσεις συγχρονισμού του πυρήνα εντός της DSR. Αυτό επιτρέπει σε μία DSR να αφυπνίσει ένα νήμα για επεξεργασία μετά την διακοπή που συνέβη. Ωστόσο, η DSR δεν πρέπει να πραγματοποιήσει μια κλήση συγχρονισμού που μπλοκάρει. Το μπλοκάρισμα εμφανίζεται όταν κατά την εκτέλεση πρέπει να περιμένει για έναν πόρο που θα αποδεσμευτεί αργότερα..

Όπως ήδη αναφέρθηκε διαφορετικές αρχιτεκτονικές υποστηρίζουν τα σήματα διακοπών με διαφορετικές μεθόδους. Το eCOS παρέχει μια τυποποιημένη μέθοδο για τον χειρισμό των διακοπών σε όλες τις αρχιτεκτονικές HAL. Η ARM αρχιτεκτονική είναι η εξαίρεση που αποκλίνει από το τυποποιημένη μοντέλο χειρισμού διακοπών.



Σχήμα 4.36 Εξυπηρέτηση διακοπών.

Ο μηχανισμός διεκπεραίωσης των διακοπών του eCOS μπορεί να παρακαμφθεί με την εγκατάσταση ενός διανύσματος ρουτινών εξυπηρέτησης στον πίνακα VSR απευθείας από το επίπεδο εφαρμογών. Στη συνέχεια η εφαρμογή είναι αποκλειστικά υπεύθυνη για την υλοποίηση , σε γλώσσα assembly, όλης της απαραίτητης υποστήριξης.

Στο σχήμα 4.36 παρουσιάζει τη ροή εκτέλεσης της μεθόδου χειρισμού διακοπών του eCOS. Στο σχήμα φαίνονται και τα επίπεδα - το επίπεδο εφαρμογής, πυρήνα και HAL - που είναι υπεύθυνα για την εκτέλεση κάθε κομμάτι του κώδικα.

Το eCOS χρησιμοποιεί ένα μηχανισμό που τον ονομάζει εικονικά διανύσματα. Τα εικονική διανύσματα είναι μια ομάδα από δείκτες σε συναρτήσεις υπηρεσιών και σε δεδομένα. Ο κύριος ρόλο τους είναι να επιτρέψουν στις υπηρεσίες που παρέχονται σε μια διαμόρφωση εκκίνησης ROM, να προσεγγιστούν από μια ρύθμιση παραμέτρων εκκίνησης RAM, για την εκσφαλμάτωση μιας εφαρμογής.

Ο πυρήνας.

Ο πυρήνας παρέχει τη λειτουργικότητα που αναμένεται σε ένα RTOS, όπως είναι οι διακοπές, ο χειρισμός των εξαιρέσεων, τα νήματα, ο χρονοπρογραμματισμός και ο συγχρονισμός. Αυτά τα λειτουργικά συστατικά στοιχεία στο eCOS είναι πλήρως διαμορφώσιμα για να καλύψουν τις ανάγκες μιας εφαρμογής. Ο πυρήνας υλοποιείται σε C++ και επιτρέπει στις εφαρμογές που υλοποιούνται σε C++ να επικοινωνούν απευθείας με τον πυρήνα (δεν υπάρχει επίσημο C++ API). Υπάρχει μια ρύθμιση για να επιτρέψετε τη χρήση ενός API του πυρήνα που υποστηρίζει C. Οι συναρτήσεις του C API του πυρήνα ορίζονται στο αρχείο kapi.h. Ο πυρήνας του eCOS υποστηρίζει επίσης διασύνδεση με το πρότυπο μITRON και το πρότυπο POSIX.

Τα κριτήρια καθόρισαν την σχεδίαση και την ανάπτυξη του πυρήνα ώστε να μπορέσει να ανταποκριθεί σε λειτουργία πραγματικού χρόνου είναι:

- Interrupt latency
- Dispatch latency
- Memory footprint
- Deterministic kernel primitives

Στο eCOS οι συναρτήσεις δεν επιστρέφουν κωδικούς σφαλμάτων για τις λειτουργίες τους. Σε ένα ενσωματωμένο σύστημα η επεξεργασία των κωδικών επιστροφής λάθους μπορεί να καταναλώσει πολύτιμους κύκλους επεξεργασίας και να απαιτήσει περισσότερη μνήμη. Σε ένα ενσωματωμένο σύστημα, συνήθως, δεν υπάρχει τρόπος για να ανανήψει από ορισμένα σφάλματα και η εφαρμογή θα πρέπει να σταματήσει.

Ο πυρήνας παρέχει επιβεβαιώσεις που μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν εντός του eCOS package. Συνήθως έχουν ενεργοποιηθεί κατά τη διάρκεια της διόρθωσης σφαλμάτων και μετά την διόρθωσή τους απενεργοποιούνται. Αυτή η προσέγγιση έχει πολλά πλεονεκτήματα, όπως τον περιορισμένη επιβάρυνση για ανίχνευση σφαλμάτων κατά την λειτουργία. Σε περίπτωση σφάλματος η εφαρμογή σταματά επιτρέποντας την άμεση εκσφαλμάτωση του προβλήματος αντί να στηρίζεται στον έλεγχο ενός κωδικού επιστροφής σφάλματος.

Τα δομικά συστατικά του πυρήνα προσφέρουν μεθόδους για να διευκολυνθεί ο εντοπισμός των σφαλμάτων. Μία τέτοια μέθοδος, ενεργοποιείται από επιλογές ρύθμισης, είναι το instrumentation του πυρήνα. Αυτό επιτρέπει στον πυρήνα να καλέσει κάποιες ρουτίνες, κάθε φορά που συμβαίνουν συγκεκριμένα γεγονότα, τα οποία κάνουν εγγραφές συμβάντων σε ένα κυκλικό buffer για ανάλυση σε μεταγενέστερο χρόνο. Τα αρχεία αυτά περιλαμβάνουν χρονοσημάνσεις, τον τύπο των αρχείων και άλλα στοιχεία που μπορούν να χρησιμοποιηθούν για τον εντοπισμό σφαλμάτων ή την ανάλυση των χρονικών απαιτήσεων των κρίσιμου-χρόνου συμβάντων του πυρήνα.

Ο χρονοπρογραμματιστής

Η καρδιά του πυρήνα του eCOS είναι ο χρονοπρογραμματιστής. Η βασική του εργασία είναι να επιλέξει το νήμα που θα εκτελεστεί, την παροχή συγχρονισμού και να ελέγχει την επίδραση μιας διακοπής στην εκτέλεση ενός νήματος.

Κατά τη διάρκεια της εκτέλεσης του κώδικα του χρονοπρογραμματιστή τα σήματα διακοπής δεν είναι απενεργοποιημένα (έτσι η χρονοκαθυστέρησή τους είναι μικρή). Ένας μετρητής στον χρονοπρογραμματιστή καθορίζει αν είναι ελεύθερος ή απενεργοποιημένος. Εάν ο μετρητής κλειδώματος είναι μη μηδενικός, ο χρονοπρογραμματισμός είναι απενεργοποιημένος, όταν επιστρέψει στο μηδέν, ο χρονοπρογραμματισμός συνεχίζεται. Η μέθοδος εξυπηρέτησης σημάτων διακοπής μέσω HAL τροποποιεί το κλείδωμα για να απενεργοποιήσει τον χρονοπρογραμματισμό κατά τη διάρκεια εκτέλεσης του ISR. Τα νήματα έχουν επίσης τη δυνατότητα να κλειδώσουν και να ξεκλειδώσουν το χρονοπρογραμματιστή. Οι λειτουργίες κλείδωμα και ξεκλείδωμα είναι ατομικές λειτουργίες και χειρίζονται από τον πυρήνα.

Το eCOS υποστηρίζει δύο διαφορετικούς χρονοπρογραμματιστές που εφαρμόζουν διαφορετικές πολιτικές. Ο πυρήνας έχει κατασκευαστεί, κατά την διαμόρφωση, με χρήση μόνο ενός χρονοπρογραμματιστή ανά πάσα στιγμή. Οι χρονοπρογραμματιστές είναι οι εξής: Πολυεπίπεδη ουρά, Bitmap.

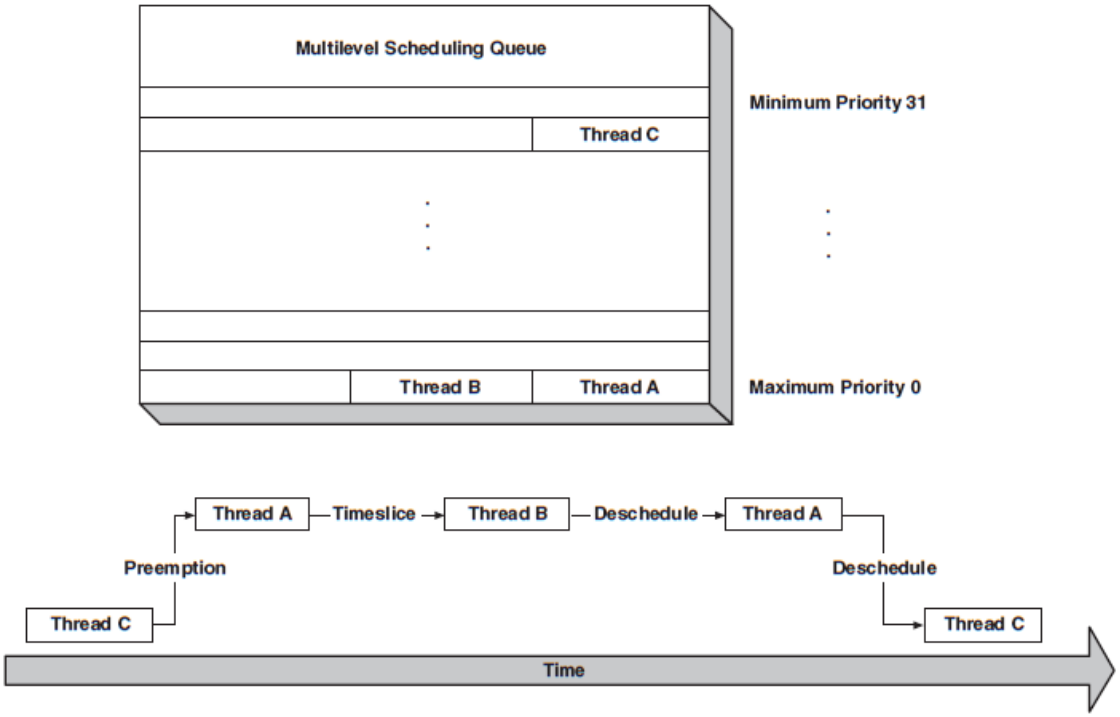
Συμμετρική πολυεπεξεργασία (SMP) υποστηρίζεται μόνο όταν χρησιμοποιείται για χρονοπρογραμματισμό η πολυεπίπεδη ουρά.

Η πολυεπίπεδη ουρά επιτρέπει την εκτέλεση πολλών νημάτων σε κάθε επίπεδο προτεραιότητας. Υπάρχουν 32 επίπεδα προτεραιότητας, από το 0 (υψηλότερη προτεραιότητα) μέχρι 31 (χαμηλότερη προτεραιότητα). Ο χρονοπρογραμματιστής επιτρέπει προεκτόπιση μεταξύ των διαφόρων επιπέδων προτεραιότητας.

Η πολυεπίπεδη ουρά επιτρέπει επίσης timeslicing μέσα σε ένα επίπεδο προτεραιότητας. Το timeslicing επιτρέπει σε κάθε νήμα σε μια προτεραιότητα να εκτελεστεί για ένα συγκεκριμένο χρόνο ο οποίος ελέγχεται από μια επιλογή διαμόρφωσης. Η υλοποίηση της ουράς για τον πολυεπίπεδο χρονοπρογραμματιστή χρησιμοποιεί διπλά συνδεδεμένες κυκλικές λίστες για να συνδέσει τα νήματα σε ένα επίπεδο προτεραιότητας και τα νήματα σε διαφορετικά επίπεδα προτεραιότητας.

Στο σχήμα 4.37 βλέπουμε την αναπαράσταση του χρονοπρογραμματισμού με πολυεπίπεδη ουρά και ένα παράδειγμα που χρησιμοποιεί αυτό το χρονοπρογραμματιστή.

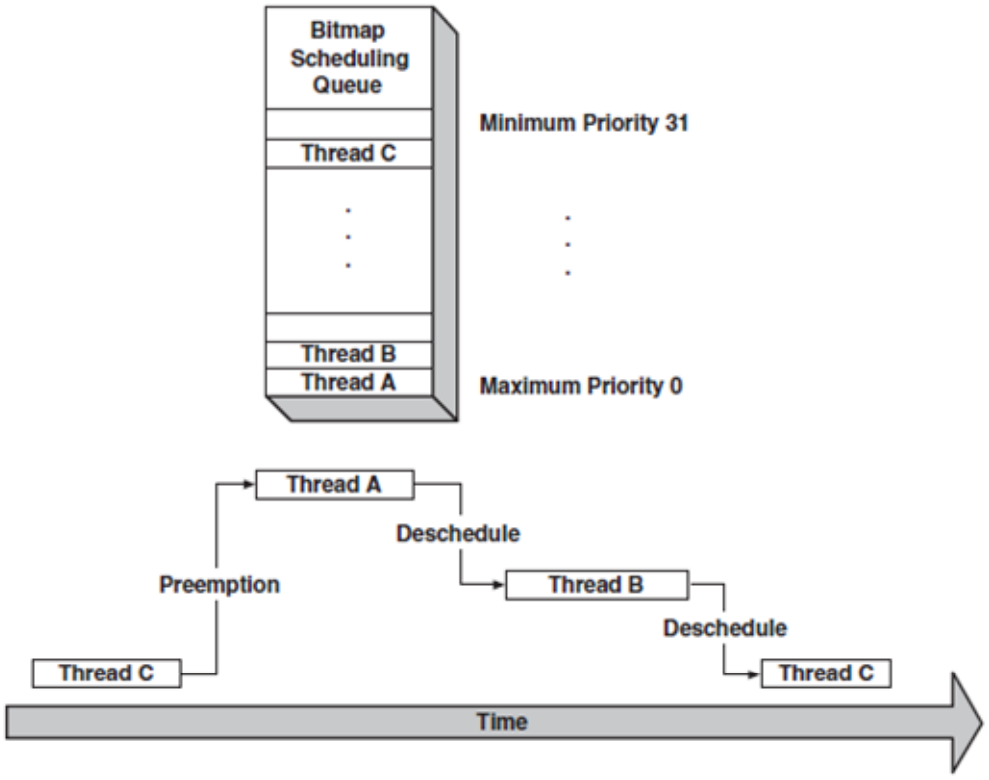
Στο σενάριο τα τρία νήματα -Thread A, B Thread, και Thread C -έχουν διαμορφωθεί κατά τη δημιουργία των νημάτων σε επίπεδα προτεραιότητας 0, 0, και 30, αντίστοιχα. Η κατάσταση της ουράς μετά τη δημιουργία του νήματος φαίνεται στο σχήμα 4.37. Για αυτό το σενάριο, η timeslicing είναι ενεργοποιημένη. Το χρονοδιάγραμμα είναι ένα στιγμιότυπο που ξεκινά με το Thread C σε εκτέλεση. Στη συνέχεια το Thread A γίνεται εκτελέσιμο, προεκτοπίζοντας το Thread C. Γίνεται μεταγωγή περιβάλλοντος. Κατά τη διάρκεια της εκτέλεσης του Thread A, το Thread B γίνεται επίσης εκτελέσιμο. Το Thread A συνεχίζεται μέχρι να λήξει το timeslice του. Στη συνέχεια γίνεται μεταγωγή περιβάλλοντος και το Thread B εκτελείται. Ολοκληρώνει εντός του timeslice του. Ο επαναχρονοπρογραμματισμός ενός νήματος μπορεί να συμβεί για διάφορους λόγους, για παράδειγμα περιμένει σε ένα mutex που δεν είναι ελεύθερο ή καθυστερεί για ένα καθορισμένο χρονικό διάστημα. Το Thread A έχει την ύψιστη προτεραιότητα από τις εργασίες που περιμένουν να εκτελεστούν, γίνεται μεταγωγή περιβάλλοντος και το Thread A αρχίζει να τρέχει. Μετά την ολοκλήρωση του Thread A νέα μεταγωγή περιβάλλοντος και εκτελείται το Thread C.



Σχήμα 4.37 Πολυεπίπεδη ουρά χρονοπρογραμματισμού.

Bitmap.

Το χρονοπρογραμματιστής bitmap επιτρέπει την εκτέλεση των νημάτων σε πολλαπλά επίπεδα προτεραιότητας με ένα και μόνο ένα νήμα σε κάθε επίπεδο προτεραιότητας. Αυτό απλοποιεί τον αλγόριθμο χρονοπρογραμματισμού και τον κάνει πολύ αποτελεσματικό. Ο αριθμός των επιπέδων προτεραιότητας είναι μια επιλογή διαμόρφωσης από



Σχήμα 4.38 Bitmap χρονοπρογραμματισμός.

1 - 32, που αντιστοιχούν σε αριθμούς προτεραιότητας 0 (υψηλότερη προτεραιότητα) μέχρι 31 (χαμηλότερη προτεραιότητα). Η ουρά είναι μία 8-, 16- ή 32-bit τιμή, ανάλογα με τα επιλεγέντα επίπεδα προτεραιότητας. Ένα bit στην ουρά αντιπροσωπεύει ένα επίπεδο προτεραιότητας. Ο χρονοπρογραμματιστής επιτρέπει προεκτοπισμό μεταξύ των διαφόρων επιπέδων προτεραιότητας. Δεδομένου ότι μόνο ένα νήμα επιτρέπεται σε κάθε επίπεδο προτεραιότητας, το timeslicing είναι άνευ σημασίας και είναι απενεργοποιημένο όταν επιλέγεται στη διαμόρφωση ο χρονοπρογραμματιστής bitmap.

Στο σχήμα 4.38 βλέπουμε την αναπαράσταση του bitmap χρονοπρογραμματισμού και ένα παράδειγμα που ton χρησιμοποιεί.

Όπως μπορούμε να δούμε στα σχήματα 4.37 και 4.38, ο bitmap είναι μια πιο απλοϊκή πολιτική χρονοπρογραμματισμού. ενώ η πολυεπίπεδη ουρά προσφέρει περισσότερες επιλογές για τη λειτουργία των νημάτων. Η απόφαση ποιος χρονοπρογραμματιστής θα χρησιμοποιηθεί εξαρτάται από τις ειδικές ανάγκες της εφαρμογής.

Νήματα και συγχρονισμός.

Πολλά νήματα μπορεί να υπάρχουν σε ένα πρόγραμμα. Σε κάθε νήμα επιτρέπετε να εκτελέσει τις δικές του δραστηριότητες στο σύστημα. Κάθε νήμα που ορίζεται στο eCOS έχει το δικό του πλαίσιο ή χώρο εργασίας για να ασκήσει τις λειτουργίες του και ένα επίπεδο προτεραιότητας για την σειρά εκτέλεσης. Είναι δουλειά του χρονοπρογραμματιστή να καθορίσει το νήμα που δικαιούται να τρέχει σε κάθε δεδομένη στιγμή. Ο πυρήνας περιέχει API συναρτήσεις για τον έλεγχο των νημάτων μέσα σε μια εφαρμογή.

Το eCOS προσφέρει επιλογές διαμόρφωσης που ελέγχουν τη συμπεριφορά των νημάτων στο σύστημα και παρέχουν πρόσθετες δυνατότητες για την υποστήριξη διαφόρων εργασιών στα νήματα.

Η εφαρμογή είναι υπεύθυνη για την παροχή της στοίβας σε ένα νήμα που χρησιμοποιείται για τις τοπικές μεταβλητές και την παρακολούθηση των κλήσεων συναρτήσεων και των επιστροφών. Η στοίβα είναι συνήθως με τη μορφή στατικών δεδομένων για να μην υπάρχει η ανάγκη για δυναμική κατανομή μνήμης. Είναι σημαντικό να γνωρίζουμε τις απαιτήσεις μεγέθους για ένα συγκεκριμένο νήμα ώστε η στοίβα που θα δημιουργηθεί να έχει το σωστό μέγεθος. Αυτό εξαλείφει τη σπατάλη μνήμης αν το μέγεθος της στοίβας που έχει τεθεί είναι πολύ μεγάλο, ή το πιο σημαντικότερο, αποφεύγει την υπερχείλιση της εάν το μέγεθος στοίβας έχει οριστεί σε πολύ μικρό. Η υπερχείλιση είναι ένα πολύ δύσκολο πρόβλημα να εντοπιστεί.

Το μέγεθος της στοίβας εξαρτάται από παράγοντες, οι οποίοι καθορίζονται από τα χαρακτηριστικά του κώδικα που εκτελούνται από το νήμα. Για παράδειγμα, πολλές εμφωλευμένες κλήσεις συναρτήσεων ή μεγάλη τοπικά διανύσματα απαιτούν μεγαλύτερα μεγέθη στοίβας για ένα συγκεκριμένο νήμα. Υπάρχουν επίσης επιλογές διαμόρφωσης που μπορεί να έχουν επίδραση στην χρήση της στοίβας ενός νήματος.

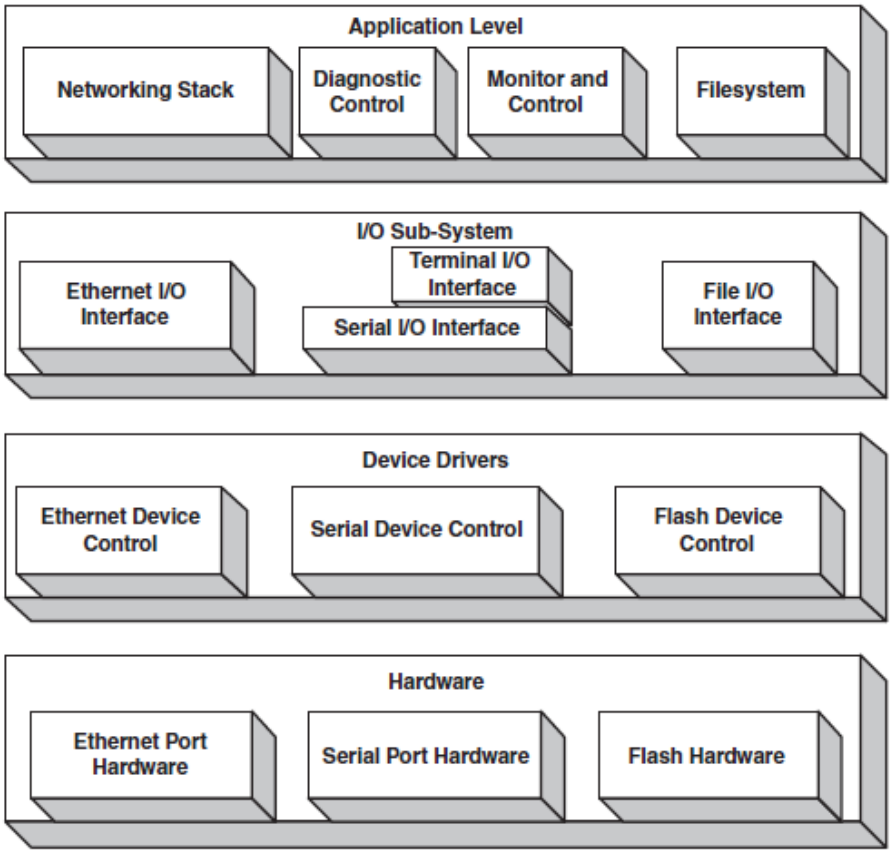
Συγχρονισμός.

Ο πυρήνας του eCOS προβλέπει μηχανισμούς για την επικοινωνία των νημάτων μεταξύ τους και τον συγχρονισμό της πρόσβασης σε κοινόχρηστους πόρους. Οι μηχανισμοί που έχει είναι οι εξής: mutexes, σηματοφορείς, condition variables, Flags, Message boxes, Spinlocks (για συστήματα SMP).

Ο πυρήνας παρέχει επίσης συναρτήσεις API που επιτρέπουν στις εφαρμογές να κάνουν χρήση των μηχανισμών του συγχρονισμού. Μερικές από αυτές είναι είτε blocking είτε nonblocking. Κλήση συναρτήσεων blocking όπως cyg_semaphore_wait, σταματούν την εκτέλεση ενός νήματος μέχρι η συνάρτηση να ολοκληρωθεί με επιτυχία. Non--blocking κλήσεις όπως cyg_semaphore_trywait, προσπαθούν να ολοκληρωθούν με επι-

τυχία, ωστόσο, εάν δεν είναι επιτυχής, ένας επιστρεφόμενος κωδικός δείχνει την κατάσταση της κλήσης, ώστε το νήμα να προχωρήσει στην εκτέλεση.

Ο πρώτος μηχανισμός συγχρονισμού που παρέχεται είναι το mutex. Το mutex, αντικείμενο αμοιβαίου αποκλεισμού, επιτρέπει σε πολλαπλά νήματα να μοιραστούν ένα πόρο σειριακά. Ο πόρος μπορεί να είναι μια περιοχή της μνήμης ή ένα κομμάτι του υλικού, όπως ο ελεγκτής άμεσης πρόσβασης στη μνήμη (DMA). Το mutex είναι παρόμοιο με ένα δυαδικό σηματοφορέα αλλά έχει μόνο δύο καταστάσεις κλειδωμένο και ξεκλείδωτο. Το mutex παρέχει προστασία από κληρονόμηση προτεραιότητας, ενώ ένας δυαδικός σηματοφορέας όχι. Επίσης το Mutex έχει ιδιοκτήτη ενώ ο σηματοφορέας όχι. Για προστασία από αντιστροφή προτεραιότητας το eCOS υποστηρίζει και το πρωτόκολλο PIP και το πρωτόκολλο PCP.



Σχήμα 4.39 Λειτουργίες I-O.

Άλλα χαρακτηριστικά.

Το eCOS:

για να βοηθήσει την εκσφαλμάτωση παρέχει δύο μηχανισμούς τα asserts και το tracing.

παρέχει συμβατότητα με την ISO C library, και δική του βιβλιοθήκη με μαθηματικές συναρτήσεις.

υποστηρίζει λειτουργίες I/O control system με δύο τρόπους ο ένας είναι με ένα I/O Sub-System και ο άλλος με Device Drivers.

Οι περισσότερες αρχιτεκτονικές επεξεργαστών παρέχουν ένα ρολόι ή μηχανισμό χρονοδιακόπτη, συνήθως ένα προγραμματιζόμενο καταχωρητή που δημιουργεί μια περιοδική διακοπή. Το eCOS να χρησιμοποιεί τον μηχανισμό χρόνου του υλικού να καθοδηγεί τα χρονικά χαρακτηριστικά του που είναι;

- Μετρητές
- Ρολόγια
- Συναγερμοί

Πλατφόρμες υλικού.

Υποστηρίζει όλες τις οικογένειες επεξεργαστών σχεδόν σε όλες τις παραλλαγές τους. Μία πλήρης λίστα των επεξεργαστών που υποστηρίζει βρίσκετε στην ιστοθέση http://ecos.sourceware.org/hardware.html [38].

- της οικογένειας x86
- της οικογένειας ARM
- της οικογένειας XScale
- της οικογένειας PowerPC
- της οικογένειας MIPS
- της οικογένειας SH-4
- της οικογένειας • Fujitsu FR-V
- της οικογένειας • Hitachi H8/300
- της οικογένειας • Matsushita AM3x
- της οικογένειας • NEC V8xx
- της οικογένειας • Samsung CalmRISC16/32
- της οικογένειας • SPARC
- της οικογένειας • SPARClite
- της οικογένειας • SuperH

5. Συμπεράσματα.

Τα τελευταία χρόνια το ενδιαφέρον για τα συστήματα πραγματικού χρόνου μεγαλώνει. Παρουσιάζονται περισσότεροι και αρτιότεροι προσομοιωτές ανοικτού λογισμικού για χρονοπρογραμματισμό πραγματικού χρόνου που μπορούν να χρησιμοποιηθούν για έρευνα και για εκπαίδευση. Σε αυτή την εργασία περιγράψαμε μερικούς προσομοιωτές με προσανατολισμό κυρίως την εκπαίδευση και παρουσιάσαμε εκτενέστερα τον Cheddar ο οποίος φαίνεται να είναι ο αρτιότερος, ο πληρέστερος και το κυριότερο άμεσα διαθέσιμος με εξαιρετική τεκμηρίωση.

Τα λειτουργικά συστήματα πραγματικού χρόνου είναι ένας χώρος με δεκάδες προτάσεις είτε εμπορικού λογισμικού είτε ελεύθερου λογισμικού. Καλύπτουν το χώρο από απλά ενσωματωμένα συστήματα με επεξεργαστή 8bit μέχρι πολύπλοκα συστήματα πραγματικού χρόνου με ένα ή περισσότερους επεξεργαστές 64bit. Ο πιο εύκολος τρόπος για να περιοριστεί το πλήθος των επιλογών που υπάρχουν είναι να μπει ένας αυστηρός περιορισμός από την πλευρά του υλικού για παράδειγμα επεξεργαστής 8bit χωρίς MMU με διαθέσιμη μνήμη 16Kbytes.

Τα δικαιώματα ανά μονάδα παραγόμενου προϊόντος και το κόστος κτήσης είναι ένας σημαντικός παράγοντας για την επιλογή ενός συγκεκριμένου προϊόντος. Στο χώρο των ήπιων συστημάτων πραγματικού χρόνου οι διάφορες παραλλαγές του linux αλλά και άλλα προϊόντα ανοικτού κώδικα φαίνεται να κυριαρχούν στην αγορά. Στα μεγάλα και αυστηρού χρόνου συστήματα η κυριαρχία των εμπορικών λειτουργικών συστημάτων πραγματικού χρόνου είναι συντριπτική. Στο χώρο των συστημάτων πραγματικού χρόνου που βασίζονται σε επεξεργαστές 8bit παραμένει δημοφιλής λύση η κατασκευή ή τροποποίηση in-house κάποιου πυρήνα πραγματικού χρόνου.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	Ελληνικός Όρος
asynchronous	ασύγχρονο
absolute deadline	απόλυτη χρονική προθεσμία
alarm	συναγερμός
best effort	βέλτιστη προσπάθεια
blocked	μπλοκαρισμένη
buffer	απομονωτής
compiler	μεταγλωττιστής
conformance class	κατηγορία συμμόρφωσης
context switch .	μεταγωγή πλαισίου
countdown timer	χρονομετρητής αντίστροφης μέτρησης
deadline	προθεσμία
debugger	πρόγραμμα εκσφαλμάτωσης, εκσφαλματωτής
dependency of tasks	αλληλοεξάρτηση διεργασιών
device driver	οδηγός συσκευής
dispatcher	αποστολέας
editor	συντάκτης
elected	επιλεγμένη
election table	πίνακας επιλογής
embedded	ενσωματωμένο
event	συμβάν
feasibility	εφικτότητα
firm	σταθερό
footprint	αποτύπωμα
fragmentation	κατακερματισμός
hard	αυστηρό
hard RTOS	αυστηρό λειτουργικό σύστημα πραγματικού χρόνου
interprocess communication	ενδοδιεργασιακή επικοινωνία
interupt	διακοπή
interupt Service Routine .	ρουτίνα εξυπηρέτησης διακοπής
interval timer	χρονομετρητής μέτρησης διανυθέντος χρόνου
kernel	πυρήνας
latency	χρόνος καθυστέρησης εξυπηρέτησης
laxity	χαλαρότητα
maximum jitter	μέγιστο χρονικό περιθώριο
memory locking	κλείδωμα μνήμης
memory map	χάρτης μνήμης
microkernel	μικροπυρήνας
middleware	μεσισμικό

multithread	πολυνηματικό
mutexes	μηχανισμούς αμοιβαίου αποκλεισμού
nanokernel	νανοπυρήνας
nested interrupt	εμφωλευμένη διακοπή
nominal laxity	ονομαστική χαλαρότητα
pools	περιοχές μνήμης
portability	μεταφερσιμότητα
predictability	προβλεψιμότητα
preemptive	προεκτοπιστικός
priority Inheritance	κληρονόμηση προτεραιότητας
priority invertion	αντιστροφή προτεραιότητας
ready	έτοιμη
real Time Operating System	λειτουργικό Σύστημα Πραγματικού Χρόνου
real Time System	σύστημα πραγματικού χρόνου
real-Time	πραγματικός χρόνος
real-time files	αρχεία πραγματικού χρόνου
Real-Time Kernel	πυρήνας πραγματικού χρόνου
Real-time thread	νήμα πραγματικού χρόνου
relative laxity	σχετική χαλαρότητα
remote Process Call .	απομακρυσμένη κλήση διεργασίας
scalability	κλιμάκωση
scheduler	χρονοπρογραμματιστής
Scheduling	χρονοπρογραμματισμός
semaphores	σηματοφορέας
shared memory	κοινόχρηστη μνήμη
soft	ήπιο
soft RTOS	χαλαρό λειτουργικό σύστημα πραγματικού χρόνου
synchronization	συγχρονισμός
synchronous I/O	σύγχρονο I/O
task	εργασία
task management	διαχείριση διεργασιών
task relative deadline	σχετική χρονική προθεσμία
task release time	χρόνος άφιξης της διεργασίας
task switch	εναλλαγή εργασίας
task Synchronization	συγχρονισμός διεργασιών
Time sharing	χρονομερισμός
timer	χρονομετρητής
tracers	ιχνηθέτες
two level scheduling	Χρονοπρογραμματισμός δύο επιπέδων
urgency	βαθμός επείγοντος

Ελληνικός Όρος	**Ξενόγλωσσος όρος**
αλληλοεξάρτηση διεργασιών	dependency of tasks
αντιστροφή προτεραιότητας	priority invertion
απόλυτη χρονική προθεσμία	absolute deadline
απομακρυσμένη κλήση διεργασίας	remote Process Call .
απομονωτής	buffer
αποστολέας	dispatcher
αποτύπωμα	footprint
αρχεία πραγματικού χρόνου	real-time files
ασύγχρονο	asynchronous
αυστηρό	hard
αυστηρό λειτουργικό σύστημα πραγματικού χρόνου	hard RTOS
βαθμός επείγοντος	urgency
βέλτιστη προσπάθεια	best effort
διακοπή	interupt
διαχείριση διεργασιών	task management
εμφωλευμένη διακοπή	nested interrupt
εναλλαγή εργασίας	task switch
ενδοδιεργασιακή επικοινωνία	interprocess communication
ενσωματωμένο	embedded
επιλεγμένη	elected
εργασία	task
έτοιμη	ready
εφικτότητα	feasibility
ήπιο	soft
ιχνηθέτες	tracers
κατακερματισμός	fragmentation
κατηγορία συμμόρφωσης	conformance class
κλείδωμα μνήμης	memory locking
κληρονόμηση προτεραιότητας	priority Inheritance
κλιμάκωση	scalability
κοινόχρηστη μνήμη	shared memory
λειτουργικό Σύστημα Πραγματικού Χρόνου	real Time Operating System
μέγιστο χρονικό περιθώριο	maximum jitter
μεσισμικό	middleware
μεταγλωττιστής	compiler
μεταγωγή πλαισίου	context switch .
μεταφερσιμότητα	portability
μηχανισμούς αμοιβαίου αποκλεισμού	mutexes
μικροπυρήνας	microkernel
μπλοκαρισμένη	blocked
νανοπυρήνας	nanokernel

νήμα πραγματικού χρόνου	Real-time thread
οδηγός συσκευής	device driver
ονομαστική χαλαρότητα	nominal laxity
περιοχές μνήμης	pools
πίνακας επιλογής	election table
πολυνηματικό	multithread
πραγματικός χρόνος	real-Time
προβλεψιμότητα	predictability
πρόγραμμα εκσφαλμάτωσης, εκσφαλματωτής	debugger
προεκτοπιστικός	preemptive
προθεσμία	deadline
πυρήνας	kernel
πυρήνας πραγματικού χρόνου	Real-Time Kernel
ρουτίνα εξυπηρέτησης διακοπής	interupt Service Routine .
σηματοφορέας	semaphores
σταθερό	firm
συγχρονισμός	synchronization
συγχρονισμός διεργασιών	task Synchronization
σύγχρονο I/O	synchronous I/O
συμβάν	event
συναγερμός	alarm
συντάκτης	editor
σύστημα πραγματικού χρόνου	real Time System
σχετική χαλαρότητα	relative laxity
σχετική χρονική προθεσμία	task relative deadline
χαλαρό λειτουργικό σύστημα πραγματικού χρόνου	soft RTOS
χαλαρότητα	laxity
χάρτης μνήμης	memory map
χρονοπρογραμματισμός δύο επιπέδων	two level scheduling
χρονομερισμός	Time sharing
χρονομετρητής	timer
χρονομετρητής αντίστροφης μέτρησης	countdown timer
χρονομετρητής μέτρησης διανυθέντος χρόνου	interval timer
χρονοπρογραμματισμός	Scheduling
χρονοπρογραμματιστής	scheduler
χρόνος άφιξης της διεργασίας	task release time
χρόνος καθυστέρησης εξυπηρέτησης	latency

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ARM	Advanced RISC Machine
ANSI	American National Standards Institute
API	Application Programming Interface
AUTRSS	AU Real-time Scheduler Simulator
AUTOSAR	AUTomotive Open System Architecture
BCC1	Basic Conformance Class 1
BIOS	Basic Inout Output System
BTRON	Business TRON
CTRON	Central and Communications TRON
CPU	Central Processing Unit
CASE	Computer-aided software engineering
DSR	Deferred Service Routine
DSP	Digital Signal Processor
EDF	Earliest Deadline First
eCos	embedded Configurable Operating System
ECC1	Extended Conformance Class 1
XML	Extensible Markup Language
FCFS	First Come First Served
FIFO	First In, First Out
GPL	General Public License
Gtk	GIMP Toolkit
Gnat	GNU compiler for the Ada programming language
GDB	GNU Debugger
GIMP	GNU Image Manipulation Program
GNU	GNU's Not Unix
HAL	Hardware Abstraction Layer
ID	Identification Data
iTRON	Industrial TRON
IDA	Instant Device Activation
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
IPSEC	Internet Protocol Security
IPv4	Internet Protocol version 4
IPC	Interprocess communication
ISR	interrupt service routine
DM	inverse Deadline ή Deadline Monotonic
JTRON	Java TRON
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench)
LIFO	last in, first out
LLF	Least Laxity First

MTRON	Macro TRON
MMU	Memory Management Unit
MIPS	Microprocessor without Interlocked Pipeline Stages
NEC	Nippon Electric Company
NTT	Nippon Telegraph and Telephone Corporation
OSEK	Offene Systeme und Deren Schnittstellen für die im Kraftfahrzeug Elektronik
OS	Operating System
OSE	Operating System Embedded
OSEck	Operating System Embedded compact kernel
OIL	OSEK Implementation Language
PDA	Personal Ddigital Assistant
PSA	Peugeot Société Anonyme
POSIX	Portable Operating System Interface
PCP	Priority Ceiling Protocol
PIP	Priority Inheritance Protocol
RM	Rate Monotonic
RMA	Rate Monotonic Analysis
RMS	Rate Monotonic Scheduling
ROM	Read Only Memory
RTLinux	Real Time Linux
RTOS	Real Time Operating System
RTSIM	Real Time simulator
RTS	Real Time System
Realtss	Real Time Systems Simulator
RTAI	Real-Time Application Interface
RR	Round Robin
SSH	Secure Shell
STRON	Silicon TRON
SRP	Stack Resource Protocol
SMP	Symmetric Multiprocessor System
TCB	Task Control Block
TRON	The Real-time Operating system Nucleus
TCL	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
VDX	Vehicle Distributed eXecutive
WCET	worst-case execution times

ΑΝΑΦΟΡΕΣ

[1] J. A. Stankovic, «Misconceptions about Real-Time Computing: A Serious Problem for Next-Generation Systems», IEEE Computer, vol. 21, no. 10, pp. 10-19, October 1988.

[2] G. C. Buttazzo, Hard Real-Time Computing Systems, Springer, 2010, p 6, p39-40.

[3] David Kalinsky, [2003]. Basic concepts of real-time operating systems.http://www.kalinskyassociates.com/Wpaper1.html [Προσπελάστηκε 01/11/11].

[4] IEEE, [1996]. Information Technology Portable Operating System Interface. (POSIX) Part 1: System Application: Program Interface (API) [C Language]. 1996, ANSI/IEEE Std 1003.1 (www.ieee.org).

[5] Baskiyar, S. «A survey of contemporary real-time operating systems». Informatica 2005, no 29, pp. 233-240.

[6] P. A. Laplante, «Criteria and an objective approach to selecting commercial real-time operating systems based on published information». Int. J. Comput. Appl. 2005, 27(2), pp 82 - 96.

[7] C. L. Liu and J. W. Layland. «Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment». Journal of the Association for Computing Machinery, 20(1), pp 46-61, January 1973.

[8] Ripoll I. et al. «Rtos state of the art analysis. Technical report», OCERA Project, 2003.

[9] N. C. Audsley, Alan Burns, M. F. Richardson, and Andy J. Wellings. «Stress: a simulator for hard real-time systems». Software - Practice and Experience, 24(6): pp 543-564, June 1994.

[10] Thorsten Kramp, Matthias Adrian, and Rainer Koster. «An open framework for real-time scheduling simulation». Lecture Notes in Computer Science, 1800: pp 766-770, January 2000.

[11] E. Manacero, M. B. Miola, and V. A. Nabuco. «Teaching real-time with a scheduler simulator». In Proceedings of 31st ASEE/IEEE Frontiers in Education Conference, pp 15-19, Reno, NV, October 2001.

[12] G. Jakovljevic, Z. Rakamaric, and D. Babic. «Java simulator of realtime scheduling algorithms». In Proceedings of the 24th International Conference on Information Technology Interfaces, volume 1, pp411-416, Croatia, June 2002.

[13] A. Diaz, R. Batista and O. Castro. «Realtss: a real-time scheduling simulator». 2007 4th International Conference on Electrical and Electronics Engineering (ICEEE 2007), Mexico City, Mexico September 2007

[14] A. Espinoza. Kiwi user guide. Technical report, Universidad Politecnica de Valencia, 2003. Available on-line at http://www.dsic.upv.es/users/ia/sma/tools/kiwi/index.html.

[15] C. Yaashuwanth and R. Ramesh. «Web-Enabled Framework for Real-Time Scheduler Simulator: A Teaching Tool», Journal of Computer Science 6 (4): pp 374-380, 2010.

[16] F. Singhoff, J. Legrand, L. Nana, L. Marc´e, «Cheddar : a Flexible Real Time Scheduling Framework», ACM SIGADA'2004 International conference Proceedings, November 14–18, 2004.

[17] IEEE, [1996]. Information Technology Portable Operating System Interface. (POSIX) Part 1: System Application: Program Interface (API) [C Language]. 1996, ANSI/IEEE Std 1003.1 (www.ieee.org).

[18] standards.ieee.org/develop/wg/POSIX.html [Προσπελάστηκε 01/11/11]

[19] www.osek-vdx.org [Προσπελάστηκε 01/11/11]

[20] OSEK/VDX Operating System Version 2.2.3 February 17th, 2005.

[21] www.autosar.org/ [Προσπελάστηκε 01/11/11].

[22] www.t-engine.org/ [Προσπελάστηκε 01/11/11].

[23] μITRON 4.0 Specification Ver. 4.03.00, TEF024-S001-04.03.00/en July 2010.

[24] www.t-engine.org/tron-project [Προσπελάστηκε 01/11/11].

[25] www.t-engine.org/tron-project/itron [Προσπελάστηκε 01/11/11].

[26] www.t-engine.org/t-kernel-2-1 [Προσπελάστηκε 01/11/11].

[27] www.enea.com/software/products/rtos/ose/ [Προσπελάστηκε 01/11/11].

[28] J.L.Boulet and A.Marchesin «Systems for small enterprises: upgrades to the Alcatel OmniOffice platform and the use of Linux», Alcatel telecommunications Review, 2001

[29] Fredrik Eriksson, «Porting OSE Systems to Linux», master's thesis, Malarden University Sweden, June 10, 2010.

[30] Muhammad Imran Mughal and Razwan Javed, «Recording of Scheduling and Communication Events on Complex Telecom Systems», master's thesis Malarden University Sweden19th June 2008

[31] www.enea.com/software/products/rtos/oseck/ [Προσπελάστηκε 01/11/11].

[32] www.qnx.com [Προσπελάστηκε 01/11/11].

[33] QNX® Neutrino® RTOS System Architecture release 6.4.1, 2009.

[34] www.qnx.com/products/neutrino-rtos/neutrino-rtos.html [Προσπελάστηκε 01/11/11].

[35] C. Chang and T. Yu, «Using Linux for real-time applications», LSKA 2010 Survey Report, March 29, 2010.

[36] http://ecos.sourceware.org/ [Προσπελάστηκε 01/11/11].

[37] Anthony J. Massa, Embedded software development with eCos, Prentice Hall, ISBN 0-13-035473-2, 2002.

[38] http://ecos.sourceware.org/hardware.html [Προσπελάστηκε 01/11/11].

[39] http://beru.univ-brest.fr/~singhoff/cheddar/ [Προσπελάστηκε 01/11/11].˝