Κεφάλαιο

Λειτουργικά Συστήματα - Διεργασίες

Σκοπός του κεφαλαίου αυτού είναι να παρουσιάσει τα Λειτουργικά Συστήματα, και να περιγράψει τις πιο βασικές οντότητες του Λειτουργικού Συστήματος, τις διεργασίες.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να εξηγείς τι είναι Λειτουργικό Σύστημα και ποιες είναι οι αρμοδιότητές του.
- Να περιγράφεις το ρόλο της διεργασίας σε ένα Λειτουργικό Σύστημα.
- Να ορίζεις το πρόβλημα του κρίσιμου τμήματος και να προτείνεις τρόπους για τη λύση του.

Μαθήματα

Μάθημα 7.1 Λειτουργικά Συστήματα

Σκοπός του μαθήματος

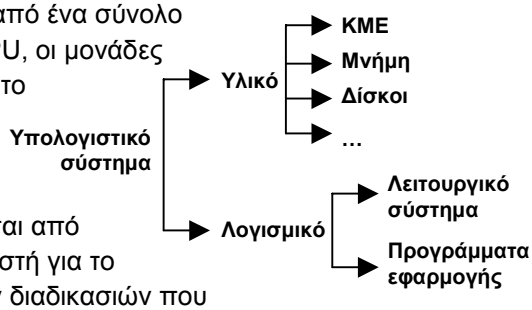
Σκοπός του μαθήματος αυτού είναι να ορίσει τα Λειτουργικά Συστήματα, να παρουσιάσει την ιστορία τους και να περιγράψει τη βασική τους δομή.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τι είναι Λειτουργικό Σύστημα*
- *Να απαριθμείς τις διάφορες κατηγορίες Λειτουργικών Συστημάτων που έχουν εμφανιστεί*
- *Να περιγράφεις τη βασική δομή ενός Λειτουργικού Συστήματος*

Όπως γνωρίζουμε, κάθε υπολογιστικό σύστημα αποτελείται (α) από ένα σύνολο συσκευών, (όπως η Κεντρική Μονάδα Επεξεργασίας - ΚΜΕ / CPU, οι μονάδες αποθήκευσης όπως μαγνητικοί και οπτικοί δίσκοι, οι εκτυπωτές, το πληκτρολόγιο κλπ), οι οποίες ονομάζονται *υλικό* του υπολογιστή (hardware), και (β) από το *λογισμικό* (software) το οποίο αποτελείται από το *Λειτουργικό Σύστημα* και τα *προγράμματα εφαρμογής*. Τα προγράμματα εφαρμογής γράφονται από τους χρήστες-προγραμματιστές και δίνουν εντολές στον υπολογιστή για το πώς θα χρησιμοποιήσει τις συσκευές για την εκτέλεση διαφόρων διαδικασιών που συνδέονται με το υπολογιστικό σύστημα.



Το Λειτουργικό Σύστημα (ΛΣ) είναι ένα σύνολο προγραμμάτων που λειτουργεί ως σύνδεσμος ανάμεσα στα προγράμματα του χρήστη και το υλικό και καθορίζει τον τρόπο λειτουργίας του υπολογιστικού συστήματος, ελέγχοντας και συντονίζοντας τη χρήση των μονάδων του από τα διάφορα προγράμματα εφαρμογής των χρηστών.

Αν φανταστούμε τον υπολογιστή ως ένα ταξί με τους χρήστες και τα προγράμματά τους να αποτελούν τους επιβάτες, τότε το ΛΣ παίζει το ρόλο του οδηγού, χωρίς τη συνεχή παρουσία του οποίου το ταξί είναι άχρηστο

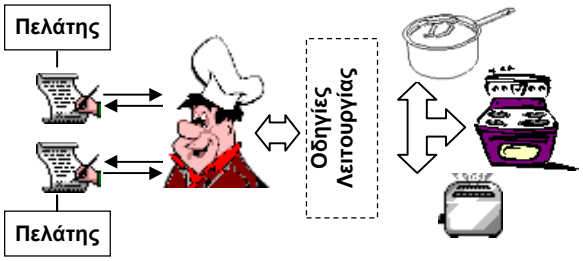
Οι στόχοι ενός ΛΣ είναι:

1. **Η διευκόλυνση των χρηστών**. Τα ΛΣ υπάρχουν επειδή κάνουν πιο εύκολη τη χρήση των υπολογιστικών συστημάτων και δίνουν τη δυνατότητα σε ανθρώπους με μικρές γνώσεις γύρω από τους υπολογιστές να εκτελέσουν πολύπλοκες εργασίες.

2. **Η διευκόλυνση των προγραμματιστών**. Χωρίς ΛΣ κάθε πρόγραμμα έπρεπε π.χ. να ελέγχει τακτικά το πληκτρολόγιο για είσοδο από το χρήστη, να γνωρίζει τις ακριβείς εντολές που πρέπει να στείλει στον εκτυπωτή για να τυπώσει κάτι ή να οργανώνει μόνο του το χώρο αποθήκευσης των δεδομένων του σε ένα σκληρό δίσκο.

3. **Η αποδοτική λειτουργία του υπολογιστικού συστήματος**, δηλαδή η όσο το δυνατόν καλύτερη χρησιμοποίηση του υλικού, ώστε να κατανέμεται καλύτερα το υπολογιστικό φορτίο. Το ΛΣ διαθέτει τη «γενική εικόνα» όλων των προγραμμάτων που πρέπει να εκτελεστούν, όλων των χρηστών του υπολογιστικού συστήματος και των αναγκών τους· έτσι, μπορεί να ρυθμίσει καλύτερα πότε και ποια προγράμματα θα εκτελεστούν κλπ.

Ο ρόλος του ΛΣ μπορεί να περιγραφεί με μια παρομοίωση από την καθημερινή ζωή, η οποία θα χρησιμοποιείται στη συνέχεια, όπου χρειάζεται, για τη διευκρίνιση διαφόρων εννοιών. Ας παρομοιώσουμε λοιπόν ένα υπολογιστικό σύστημα με ένα ζαχαροπλαστείο το οποίο παρασκευάζει γλυκά κατά παραγγελία και σύμφωνα με τις οδηγίες που δίνουν οι πελάτες. Ο υπάλληλος του ζαχαροπλαστείου παίζει το ρόλο της Κεντρικής Μονάδας Επεξεργασίας (ΚΜΕ, CPU), και τα σκεύη του είναι οι υπόλοιπες συσκευές του υπολογιστικού συστήματος. Οι οδηγίες για την παρασκευή των γλυκών είναι τα προγράμματα και οι πελάτες είναι οι χρήστες του υπολογιστικού συστήματος. Οι οδηγίες που ακολουθεί ο υπάλληλος του ζαχαροπλαστείου για την εξυπηρέτηση των πελατών αντιστοιχούν στο ΛΣ του υπολογιστικού συστήματος.



Λειτουργικά Συστήματα Ομαδικής Επεξεργασίας

Ένας πρώτος τρόπος λειτουργίας του ζαχαροπλαστείου είναι οι πελάτες να αναμένουν σε μια ουρά και να δίνουν με τη σειρά στον ζαχαροπλάστη τις οδηγίες τους για την παρασκευή των γλυκών της προτίμησής τους. Μόνο όταν ολοκληρωθεί η εργασία για ένα πελάτη εξυπηρετείται ο επόμενος. Μια καλύτερη λύση είναι να ομαδοποιούνται πελάτες που θέλουν το ίδιο γλυκό και να εξυπηρετούνται διαδοχικά ώστε να γίνεται καλύτερη χρήση των σκευών.

Αυτός ο τρόπος εξυπηρέτησης αντιστοιχεί στα ΛΣ πρώτης γενιάς της δεκαετίας του ’50, τα *Λειτουργικά Συστήματα Ομαδικής Επεξεργασίας* (batch processing). Στα συστήματα αυτά ο χειριστής του υπολογιστικού συστήματος ομαδοποιούσε τα προγράμματα που υπέβαλλαν οι χρήστες· έτσι το ΛΣ δεχόταν μια ομάδα ομοειδών προγραμμάτων (π.χ. προγραμμάτων COBOL), τα επεξεργαζόταν το ένα μετά το άλλο (με τη σειρά εμφανίσεως) και τύπωνε τα αποτελέσματά τους πάλι με την ίδια σειρά. Τα πρώτα αυτά λειτουργικά συστήματα, παρουσίαζαν τα ακόλουθα βασικά μειονεκτήματα:

♦ **Υποαπασχόληση των συσκευών**: Σε ένα ΛΣ ομαδικής επεξεργασίας δεν ήταν δυνατή η ταυτόχρονη εκτέλεση περισσοτέρων του ενός προγραμμάτων. Έπρεπε πρώτα να τελειώσει ένα πρόγραμμα, για να αρχίσει η εκτέλεση του επομένου προγράμματος της ομάδας. Ένα πρόγραμμα όμως δεν μπορεί να απασχολεί ταυτόχρονα όλες τις μονάδες του Η/Υ, οι οποίες κατά συνέπεια υποαπασχολούνταν· για παράδειγμα, η ΚΜΕ απασχολούνταν συνήθως σε ποσοστό μικρότερο του 10%.

Εάν ήταν δυνατό να εκτελούνται πολλά προγράμματα ταυτόχρονα, θα μπορούσε σε μια δεδομένη χρονική στιγμή κάθε ένα από αυτά να απασχολεί και διαφορετική μονάδα. Έτσι θα περιοριζόταν ο ανενεργός χρόνος κάποιων μονάδων του υπολογιστή.

♦ Το χρονικό διάστημα που μεσολαβεί από τη στιγμή που ο προγραμματιστής θα αφήσει στον υπολογιστή το πρόγραμμά του μέχρι τη στιγμή που θα πάρει τα αποτελέσματα, ονομάζεται *χρόνος ανακύκλωσης* (turnaround time) και στα ΛΣ ομαδικής επεξεργασίας το διάστημα αυτό ήταν πολύ μεγάλο. Συνήθως τα αποτελέσματα των προγραμμάτων δεν ήταν διαθέσιμα, παρά μόνο αφού είχε ολοκληρωθεί η εκτέλεση όλης της ομάδας. Έτσι ο χρόνος ανακύκλωσης εξαρτιόταν από το χρόνο εκτέλεσης ολόκληρης της ομάδας, ο οποίος μπορεί να είναι π.χ. 50 ή 100 φορές μεγαλύτερος από το χρόνο εκτέλεσης του συγκεκριμένου προγράμματος.

Λειτουργικά Συστήματα Πολυπρογραμματισμού

> Η εξυπηρέτηση των πελατών του ζαχαροπλαστείου θα ήταν πιο γρήγορη αν ο υπάλληλος είχε οδηγίες από τον ιδιοκτήτη του ζαχαροπλαστείου να παρασκευάζει περισσότερα από ένα γλυκά ταυτοχρόνως. Κατά τη διάρκεια π.χ. του ψησίματος ενός γλυκού έχει τη δυνατότητα να προετοιμάζει στον αναμίκτη (μίξερ) ένα άλλο, εκμεταλλευόμενος το χρόνο του. Όταν όμως ολοκληρωθεί το ψήσιμο του πρώτου θα τον ειδοποιεί π.χ. ένα κουδουνάκι να διακόψει την προετοιμασία του δευτέρου και να συνεχίσει την παρασκευή του πρώτου γλυκού με το επόμενο στάδιό της.

Αυτή την ιδέα οργάνωσης ακολουθούν τα ΛΣ δεύτερης γενιάς (δεκαετίας ’60), τα *Λειτουργικά Συστήματα Πολυπρογραμματισμού* (multiprogramming), στα οποία επιδιώκεται η μείωση του άεργου χρόνου των μονάδων του υπολογιστή και του χρόνου ανακύκλωσης. Η διακοπή μιας λειτουργίας (το κουδουνάκι του ζαχαροπλαστείου) γίνεται με τη δημιουργία ειδικών σημάτων, των *σημάτων διακοπής* (interrupts), τα οποία διακόπτουν την τρέχουσα λειτουργία της ΚΜΕ.

Πολλές φορές το ΛΣ Πολυπρογραμματισμού ορίζεται ως το ΛΣ που επιτρέπει σε περισσότερα από ένα προγράμματα να είναι «φορτωμένα» στη μνήμη τού υπολογιστή και να εκτελούνται συγχρόνως. Φυσικά μόνο ένα πρόγραμμα μπορεί να απασχολεί ανά πάσα στιγμή κάθε μονάδα· π.χ. ένα πρόγραμμα εκτελείται από την ΚΜΕ, ένα άλλο διαβάζει από το δίσκο, ένα τρίτο στέλνει δεδομένα στη σειριακή θύρα εξόδου κλπ.

> Μια πολύ πρακτική υπηρεσία για τους πελάτες του ζαχαροπλαστείου είναι οι τηλεφωνικές παραγγελίες και η κατ’ οίκον διανομή των γλυκών, οι οποίες αντιστοιχούν στην τηλεπεξεργασία που θα δούμε αμέσως μετά.

Παράλληλα με τα ΛΣ Πολυπρογραμματισμού αναπτύχθηκαν και συσκευές οι οποίες έδιναν τη δυνατότητα σε περιφερειακά όπως τηλέτυπα, οθόνες, εκτυπωτές κλπ. να επικοινωνήσουν με ένα υπολογιστή μέσω τηλεπικοινωνιακών γραμμών. Έτσι οι περιφερειακές μονάδες δεν ήταν πλέον απαραίτητο να βρίσκονται στον ίδιο φυσικό χώρο με τον υπολογιστή. Τα ΛΣ που διαχειρίζονταν τέτοια υπολογιστικά συστήματα με «απομακρυσμένα» περιφερειακά ονομάζονται *Λειτουργικά Συστήματα Τηλεπεξεργασίας*. Παρ’ όλα τα πλεονεκτήματα που παρέχουν τα ΛΣ Πολυπρογραμματισμού, παρουσιάζουν και μειονεκτήματα. Ένα απ’ αυτά είναι π.χ. ότι το ίδιο το ΛΣ είναι πλέον ένα πολύπλοκο πρόγραμμα (σε αντίθεση με τα ΛΣ ομαδικής επεξεργασίας που είναι πολύ απλά), το οποίο με τη σειρά του απασχολεί τον

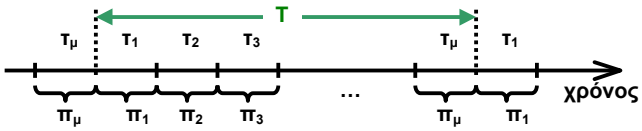
υπολογιστή σε βάρος των προγραμμάτων εφαρμογών, και μάλιστα τόσο πιο πολύ, όσο πιο πολύπλοκο είναι.

Λειτουργικά Συστήματα Καταμερισμού Χρόνου

Τερματικό
Συνδυασμός οθόνης και πληκτρολογίου για κάθε χρήστη ενός μεγάλου υπολογιστή

Πολλά μεγάλα υπολογιστικά συστήματα εξυπηρετούν πολλούς χρήστες, κάθε ένας από τους οποίους έχει στη διάθεσή του ένα τερματικό. Ο χρήστης πληκτρολογεί εντολές στο τερματικό, μέσω του οποίου διαβιβάζονται και εκτελούνται στο υπολογιστικό σύστημα. Για να έχουν όλοι οι χρήστες την εντύπωση ότι εξυπηρετούνται παράλληλα, το υπολογιστικό σύστημα καταμερίζει το χρόνο του, δίνοντας από λίγο και εκ περιτροπής στον καθένα.

> Μια άλλη μέθοδος λειτουργίας του ζαχαροπλαστείου θα ήταν η «ταυτόχρονη» παρασκευή των γλυκών, με καταμερισμό του χρόνου του υπαλλήλου μεταξύ των διαφόρων συνταγών. Κάθε πέντε λεπτά χτυπά ένα ρολόι και ο ζαχαροπλάστης διακόπτει την εργασία του, βάζει στην άκρη το γλυκό που έφτιαχνε και καταπιάνεται με το επόμενο στη σειρά.



Στα υπολογιστικά συστήματα το ρόλο του ρολογιού έχει ένα ειδικό κύκλωμα, ο *χρονιστής* (timer), το οποίο σε τακτά χρονικά διαστήματα δημιουργεί ένα σήμα διακοπής. Ο χρόνος χωρίζεται σε διαστήματα διάρκειας Τ. Αν υπάρχουν μ προγράμματα για εκτέλεση, που συμβολίζονται με $\pi_1,\pi_2,\ldots,\pi_\mu$, η περίοδος Τ διαιρείται σε μ χρονικά διαστήματα $\tau_1,\tau_2,\ldots,\tau_\mu$. Ο χρονιστής δημιουργεί ένα σήμα στην αρχή κάθε περιόδου τ_v, οπότε αρχίζει να εξυπηρετείται από την ΚΜΕ το ν-οστό πρόγραμμα. Αν κάποιο από αυτά δεν ζητά εξυπηρέτηση ή έχει τελειώσει, η σειρά του δίνεται στο επόμενο.

Ένα ΛΣ που οργανώνει τη λειτουργία του υπολογιστή κατ' αυτό τον τρόπο ονομάζεται *Λειτουργικό Σύστημα Καταμερισμού Χρόνου* (time sharing).

Η περίοδος Τ μπορεί να είναι της τάξης των 3 sec· αν υποθέσουμε ότι υπάρχουν 6 προγράμματα για εκτέλεση, στο καθένα (σε κάθε περίοδο) διατίθεται 0,5 sec, χρόνος αρκετά μεγάλος για να καλύψει πολλές φορές ολόκληρη την εκτέλεση ενός προγράμματος.

Εάν κάποιο πρόγραμμα δεν καλύπτεται, περιμένει μέχρι την επόμενη χρονική περίοδο. Μερικές φορές βέβαια ένα πρόγραμμα πρέπει να εκτελεστεί αμέσως, χωρίς καθυστέρηση, επειδή η ταχύτητα απόκρισής του έχει ουσιαστική σημασία. Τότε, σε περιπτώσεις δηλαδή λειτουργίας σε *πραγματικό χρόνο* (real-time operation), το πρόγραμμα έχει προτεραιότητα έναντι των άλλων και εξυπηρετείται πρώτο.

Λειτουργικά Συστήματα 3ης και 4ης γενιάς

Σε πολλά νεότερα ΛΣ, αυτά της *3ης γενιάς*, γίνεται συνδυασμός των παραπάνω μεθόδων, για να υποστηρίζουν ταυτόχρονα ομαδική επεξεργασία και καταμερισμό χρόνου. Η έννοια της ομαδικής επεξεργασίας έχει σήμερα διαφοροποιηθεί και καθορίζεται βασικά από την έλλειψη αλληλεπίδρασης μεταξύ του χρήστη και του προγράμματός του.

Η ανάπτυξη νέων αρχιτεκτονικών για το υλικό, όπως π.χ. παράλληλων υπολογιστών, κατανεμημένων συστημάτων καθώς και δικτύων υπολογιστών, έδωσε την αφορμή για την ανάπτυξη ακόμα πιο εξελιγμένων και πολύπλοκων ΛΣ, αυτών της *4ης γενιάς*. Στα ΛΣ αυτά γίνεται πραγματικά ταυτόχρονη εκτέλεση πολλών προγραμμάτων, αφού κάθε επεξεργαστής μπορεί να ασχολείται με ένα διαφορετικό πρόγραμμα.

Τα συστήματα πολυεπεξεργαστών αντιστοιχούν στην ύπαρξη πολλών ζαχαροπλαστών που μοιράζονται τα ίδια σκεύη· τα κατανεμημένα συστήματα αντιστοιχούν σε πολλούς υπαλλήλους, καθένα με τα δικά του σκεύη και τέλος τα δίκτυα υπολογιστών είναι αντίστοιχα με μια αλυσίδα συνεργαζομένων ζαχαροπλαστείων.

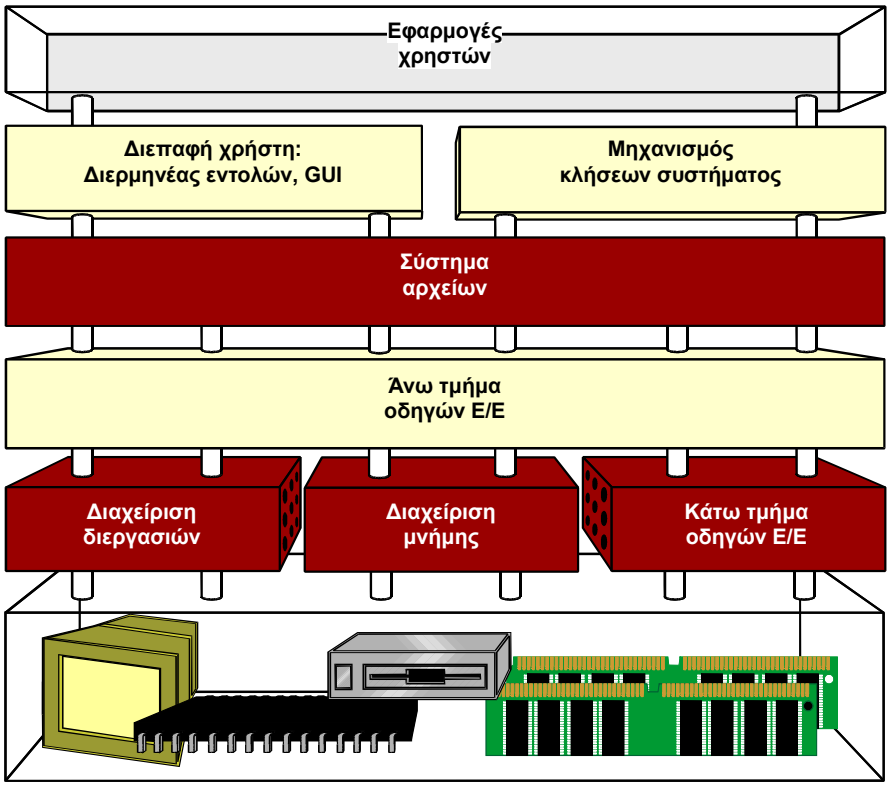
Στα υπολογιστικά συστήματα, το πρόγραμμα του ΛΣ συνήθως πωλείται από τον κατασκευαστή μαζί με το υλικό και είναι καθοριστικής σημασίας για τη λειτουργία του. Είναι δυνατόν ο ίδιος υπολογιστής να χρησιμοποιεί κατά καιρούς διαφορετικά ΛΣ, ανάλογα με τις ανάγκες των χρηστών του, και να έχει τελείως διαφορετική συμπεριφορά και απόδοση. Ακόμα, είναι δυνατόν υπολογιστές με διαφορετικό υλικό να χρησιμοποιούν το ίδιο ΛΣ και να έχουν παρόμοια συμπεριφορά, πιθανώς με διαφορετική απόδοση.

Τα ΛΣ είναι γραμμένα σε συμβολική γλώσσα ή σε συνδυασμό συμβολικής γλώσσας και κάποιας γλώσσας υψηλού επιπέδου (όπως η C).

Δομή ενός Λειτουργικού Συστήματος

Τα περισσότερα ΛΣ, και ιδιαίτερα τα σύγχρονα, είναι οργανωμένα σε *επίπεδα* (layers). Αυτό σημαίνει ότι κατά τη σχεδίασή τους έχουν διαιρεθεί σε τμήματα, και κάθε τμήμα τους επικοινωνεί μόνο με αυτά που βρίσκονται στο αμέσως ανώτερο ή το αμέσως κατώτερο επίπεδο. Όσα τμήματα χρησιμοποιούν απευθείας το υλικό του υπολογιστή, βρίσκονται στο κατώτερο επίπεδο του ΛΣ. Τα υπόλοιπα τμήματα, που βρίσκονται σε ανώτερα επίπεδα, δεν επικοινωνούν καθόλου με το υλικό, αλλά χρησιμοποιούν τα τμήματα που ανήκουν στο αμέσως κατώτερο επίπεδο.

Στο σχήμα φαίνεται ένα παράδειγμα οργάνωσης ΛΣ σε επίπεδα. Η οργάνωση αυτή βέβαια είναι ενδεικτική, γιατί υπάρχουν πολλές παραλλαγές της, αλλά η βασική φιλοσοφία είναι κοινή.



Στο χαμηλότερο επίπεδο ανήκουν τα τμήματα του ΛΣ που διαχειρίζονται:

- ☑ τη μνήμη
- ☑ τα υπό εκτέλεση προγράμματα
- ☑ τις λειτουργίες επικοινωνίας με τις περιφερειακές συσκευές.

Οι λειτουργίες αυτές αποτελούν το κάτω τμήμα των *οδηγών συσκευών E/E* (device drivers), ειδικών τμημάτων του ΛΣ που αναλαμβάνουν την επικοινωνία του ΛΣ με τα περιφερειακά. Στο αμέσως ανώτερο επίπεδο βρίσκεται το άνω τμήμα

των οδηγών Ε/Ε και πάνω από αυτό βρίσκεται το επίπεδο που κάνει τη διαχείριση του συστήματος αρχείων. Τα προγράμματα των χρηστών επικοινωνούν μόνο με το υψηλότερο επίπεδο σε ένα ΛΣ, που αποτελείται από τη *διεπαφή χρήστη* (user interface) και τις κλήσεις συστήματος. Η διεπαφή με το χρήστη μπορεί να γίνεται είτε με εντολές, με το *διερμηνέα εντολών* (command interpreter), ή με μία *διεπαφή χρήστη με χρήση γραφικών* (Graphical User Interface, GUI).

Παρατηρήστε ότι όπως είδαμε γενικά για τον υπολογιστή στο μάθημα 1.1, και το λειτουργικό σύστημα έχει ιεραρχική οργάνωση.

Ανακεφαλαίωση

Το Λειτουργικό Σύστημα είναι ένα σύνολο προγραμμάτων που οργανώνουν και επιβλέπουν τις λειτουργίες του υλικού σε ένα υπολογιστικό σύστημα. Κατά την εξέλιξή τους έχουν εμφανιστεί διάφορες κατηγορίες ΛΣ, όπως τα ΛΣ ομαδικής επεξεργασίας (1η γενιά), πολυπρογραμματισμού και καταμερισμού χρόνου (2η γενιά), πολυεπεξεργασίας (3η και 4η γενιά) κλπ. Τα περισσότερα ΛΣ είναι οργανωμένα σε επίπεδα, με το κατώτερο επίπεδο μόνο να επικοινωνεί απευθείας με το υλικό του υπολογιστή, και τα προγράμματα του χρήστη να χρησιμοποιούν μόνο το ανώτερο επίπεδο.

Γλωσσάριο όρων

Διεπαφή Χρήστη	*User Interface*
Διεπαφή Χρήστη Με Χρήση Γραφικών	*Graphical User Interface - GUI*
Διερμηνέας Εντολών	*Command Interpreter*
Επίπεδο	*Layer*
Καταμερισμός Χρόνου	*Time Sharing*
Κεντρική Μονάδα Επεξεργασίας – ΚΜΕ	*Central Processing Unit - CPU*
Λειτουργία Πραγματικού Χρόνου	*Real-Time Operation*
Λειτουργικό Σύστημα	*Operating System*
Λογισμικό	*Software*
Οδηγός Συσκευής	*Device Driver*
Ομαδική Επεξεργασία	*Batch Processing*
Πολυπρογραμματισμός	*Multiprogramming*
Πρόγραμμα Εφαρμογής	*Application Program*
Σήμα Διακοπής	*Interrupt*
Τηλεπεξεργασία	*Remote Processing*
Υλικό	*Hardware*
Χρονιστής	*Timer*
Χρόνος Ανακύκλωσης	*Turnaround Time*

Ερωτήσεις

? Ποια είναι τα βασικά τμήματα ενός υπολογιστικού συστήματος;

? Ποιο είναι το έργο ενός Λειτουργικού Συστήματος; Τι θα γινόταν αν δεν υπήρχαν ΛΣ;

? Πόσες γενιές ΛΣ έχουν εμφανιστεί, και τι είδους ΛΣ αποτελούν την κάθε γενιά;

? Περίγραψε την οργάνωση ενός ΛΣ σε επίπεδα. Τι πλεονεκτήματα νομίζεις ότι προσφέρει η οργάνωση αυτή;

Μάθημα

7.2

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει την έννοια της διεργασίας, που είναι πολύ βασική στα ΛΣ, την έννοια της ελαφριάς διεργασίας και να δείξει τις ομοιότητες και τις διαφορές τους.

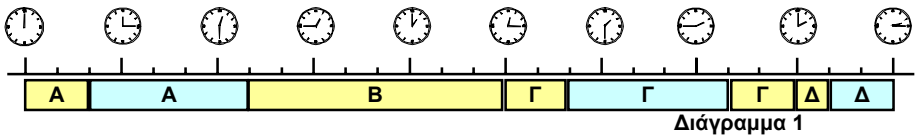
Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να περιγράφεις πώς ένα ΛΣ μπορεί να κατανέμει το χρόνο του μεταξύ πολλών προγραμμάτων*
- *Να εξηγείς τι είναι η διεργασία και σε τι διαφέρει από ένα πρόγραμμα*
- *Να ορίζεις τι είναι η μεταγωγή περιβάλλοντος*
- *Να εξηγείς τι είναι η ελαφρή διεργασία και σε τι διαφέρει από μία διεργασία*
- *Να προσδιορίζεις πότε είναι προτιμότερο να χρησιμοποιούμε τις ελαφρές διεργασίες*

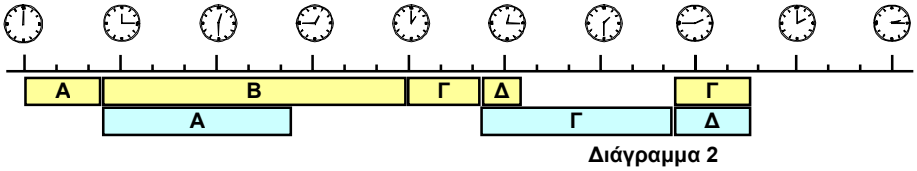
Στους περισσότερους υπολογιστές που έχουν μια μόνο ΚΜΕ, σε κάθε χρονική στιγμή μπορεί να εκτελείται μια μόνο εντολή γλώσσας μηχανής, άρα και ένα μόνο πρόγραμμα. Είναι όμως δυνατόν να κατανέμει η ΚΜΕ το χρόνο της, δίνοντας εκ περιτροπής ένα μικρό ποσοστό του χρόνου της σε κάθε ένα πρόγραμμα έτσι ώστε να δίνει την εντύπωση στους χρήστες ότι τα εκτελεί όλα μαζί.

Ας υποθέσουμε ότι ο ζαχαροπλάστης (η ΚΜΕ) έχει να παρασκευάσει τέσσερα γλυκά, και οι οδηγίες τους αναφέρουν ότι απαιτούν: το Α 35’ (10’ προετοιμασία και 25’ ψήσιμο), το Β 40’, το Γ 45’ (10’ προετοιμασία, 25’ ψήσιμο και 10’ ολοκλήρωση) και το Δ 15’ (5’ προετοιμασία και 10’ ψήσιμο) αντίστοιχα. Αν η εκτέλεση των συνταγών γίνει με τη σειρά παραλαβής τους, χωρίς παράλληλη χρησιμοποίηση του φούρνου, το όλο έργο μπορεί να οργανωθεί στις φάσεις του Διαγράμματος 1.



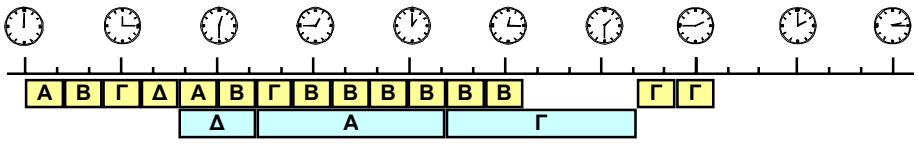
Διάγραμμα 1

Αν η εκτέλεση των συνταγών γίνεται με τη σειρά παραλαβής, αλλά με παράλληλη προετοιμασία και χρήση του φούρνου, έχουμε σημαντική βελτίωση στον απαιτούμενο χρόνο, όπως βλέπουμε στο Διάγραμμα 2.



Διάγραμμα 2

Αν όμως ο ζαχαροπλάστης αφιερώνει 5’ σε κάθε γλυκό, τότε το πρόγραμμα παρασκευής τους μπορεί να οργανωθεί σαν το Διάγραμμα 3.



Διάγραμμα 3

Ο τρόπος αυτός παρασκευής των γλυκών έχει ως αποτέλεσμα τη γρηγορότερη ολοκλήρωση αυτών που παρασκευάζονται εύκολα, ανεξάρτητα από τη σειρά με την οποία δόθηκαν οι συνταγές.

Αντίστοιχα ένα λειτουργικό σύστημα καταμερισμού χρόνου θα μοιράζει το χρόνο της ΚΜΕ σε μικρά στοιχειώδη διαστήματα, τα *κβάντα χρόνου* (time quanta). Σε κάθε κβάντο επιλέγεται ένα από τα προγράμματα και εκτελούνται εντολές από αυτό μέχρι να τελειώσει το χρονικό διάστημα που του διατέθηκε. Όταν ο χρόνος ολοκληρωθεί, επιλέγεται ένα άλλο πρόγραμμα για εκτέλεση κ.ο.κ.

Η διαδικασία αυτή είναι αρκετά πολύπλοκη και επιβαρύνει τις απαιτήσεις του ίδιου του ΛΣ. Το όφελος όμως από αυτή είναι (1) **η αύξηση της απόδοσης**, αφού το υλικό χρησιμοποιείται παράλληλα και (2) **η μείωση του χρόνου ανακύκλωσης**, αφού τα προγράμματα κατά μέσο όρο εξυπηρετούνται πιο γρήγορα.

Η συνεχής εναλλαγή ανάμεσα σε πολλές συνταγές είναι πολύ εύκολο να προκαλέσει σφάλματα. Για να μη γίνονται λοιπόν λάθη, πρέπει ο ζαχαροπλάστης κατά την εναλλαγή από τη μια συνταγή στην άλλη να κάνει ορισμένες ενέργειες όπως:

- Να σημειώνει μέχρι ποιο σημείο της τρέχουσας συνταγής έχει φθάσει, ώστε να μπορεί να συνεχίσει από το σωστό σημείο όταν έλθει ξανά η σειρά της.
- Να τοποθετήσει ό,τι από το γλυκό έχει παρασκευάσει μέχρι εκείνη τη στιγμή σε ένα ορισμένο μέρος, για να μπορεί αργότερα να το ξαναπάρει.
- Αν κάποιο βήμα της τρέχουσας συνταγής δεν είναι δυνατόν να διακοπεί, όπως π.χ. όταν κάτι ψήνεται στο φούρνο, πρέπει να σημειώσει ότι αυτό το βήμα εκκρεμεί. Όταν το ψήσιμο ολοκληρωθεί, θα τοποθετήσει το ψημένο γλυκό μαζί με τα υπόλοιπα υλικά που θα ολοκληρώσουν την παρασκευή του γλυκού και θα βάλει στο φούρνο ό,τι από την επόμενη συνταγή μέχρι τώρα βρισκόταν σε αναμονή.

Όπως στο παράδειγμα του ζαχαροπλαστείου, έτσι και στα υπολογιστικά συστήματα πρέπει να υπάρχει ένα σύνολο πληροφοριών που περιγράφουν την κατάσταση στην οποία βρίσκεται το πρόγραμμα που εκτελείται, για να μπορεί να συνεχιστεί η εκτέλεσή του από το σωστό σημείο την επόμενη φορά που θα έρθει η σειρά του.

Βλέπουμε λοιπόν ότι ένα ΛΣ διαχειρίζεται προγράμματα σε εκτέλεση· μπορεί π.χ. να εκτελεί πολλές φορές το ίδιο πρόγραμμα, μία φορά για κάθε χρήστη. Τα εκτελούμενα προγράμματα αποτελούν ανεξάρτητες οντότητες για το ΛΣ και ονομάζονται *διεργασίες* (processes).

Μια διεργασία είναι ένα πρόγραμμα ή ένα αυτόνομο τμήμα προγράμματος υπό εκτέλεση. Οι όροι *πρόγραμμα* και *διεργασία* διαφοροποιούνται από το γεγονός ότι το πρόγραμμα είναι παθητική οντότητα ενώ η διεργασία είναι ενεργητική.

Η εναλλαγή από τη μια διεργασία στην άλλη ονομάζεται *μεταγωγή περιβάλλοντος* (context switching). Οι πληροφορίες που κρατούνται κατά τη μεταγωγή περιβάλλοντος αφορούν τα ακόλουθα:

- Από ποια εντολή του προγράμματος πρέπει να συνεχισθεί η εκτέλεση της διεργασίας την επόμενη φορά
- Ποια είναι η κατάσταση της ΚΜΕ, ώστε να επαναφερθεί για να συνεχιστεί σωστά η εκτέλεση της διεργασίας
- Ποια είναι τα ενδιάμεσα αποτελέσματα που έχουν υπολογιστεί μέχρι εκείνη τη στιγμή

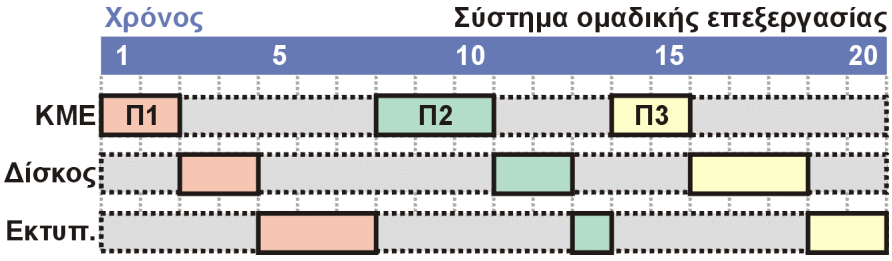
Το ΛΣ καταγράφει αυτές (και άλλες) πληροφορίες για κάθε διεργασία σε μια ειδική περιοχή της μνήμης που ονομάζεται *Σύνολο Ελέγχου Διεργασίας* (Process Control Block), ή αλλιώς *ΣΕΔ* (PCB).

Σύγκριση επίδοσης λειτουργικών συστημάτων

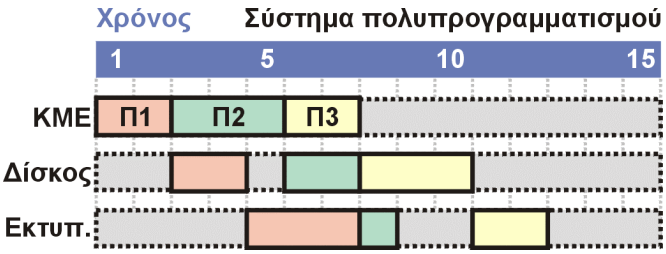
Ένα απλοποιημένο παράδειγμα μπορεί να δείξει με παραστατικό τρόπο τις διαφορές μεταξύ των διαφόρων ΛΣ και τα πλεονεκτήματα του καταμερισμού χρόνου. Σε ένα υπολογιστικό σύστημα δίνονται για εκτέλεση τρία προγράμματα με απαιτήσεις σε χρόνο που φαίνονται στον παράπλευρο πίνακα. Κάνουμε την υπόθεση ότι κάθε διεργασία χρησιμοποιεί κατά σειρά πρώτα την ΚΜΕ, μετά το δίσκο και μετά τον εκτυπωτή.

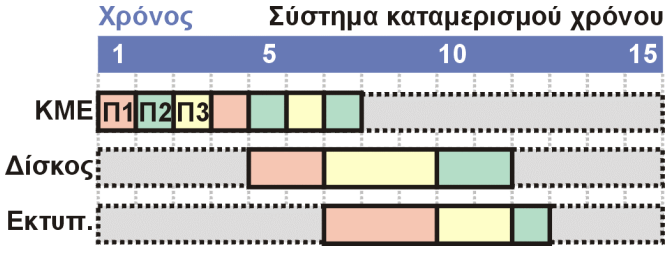
	ΚΜΕ	Δίσκος	Εκτυπωτής	Σύνολο
Πρόγ.1	2	2	3	7
Πρόγ.2	3	2	1	6
Πρόγ.3	2	3	2	7
Σύνολο	7	7	6	20

Το σχεδιάγραμμα εκτέλεσης για ένα ΛΣ ομαδικής επεξεργασίας είναι το εξής:



Για το σύστημα πολυπρογραμματισμού και το σύστημα καταμερισμού χρόνου θα έχουμε:





	Συνολικός χρόνος	Ανενεργός χρόνος	Τέλος της π_1	Τέλος της π_2	Τέλος της π_3
Ομαδική Επεξεργασία	20	40	7	13	20
Πολυπρογραμματισμός	12	16	7	8	12
Καταμερισμός χρόνου	12	16	9	12	11

Τα αποτελέσματα για τα τρια συστήματα συνοψίζονται στον παράπλευρο πίνακα. Παρατηρούμε ότι ο συνολικός χρόνος εκτέλεσης και ο ανενεργός χρόνος για το σύστημα ομαδικής επεξεργασίας είναι πολύ μεγαλύτεροι από ό,τι στα άλλα δυο συστήματα. Το σύστημα καταμερισμού χρόνου έχει άλλο ένα πλεονέκτημα: καμία διεργασία δεν ευνοείται ως προς το πότε θα τερματίσει (ενώ στο σύστημα πολυπρογραμματισμού υπάρχει τέτοια ανισότητα)· μια και όλες οι διεργασίες έχουν περίπου τις ίδιες χρονικές απαιτήσεις, τελειώνουν περίπου στον ίδιο χρόνο.

Οι επιδόσεις της εκτέλεσης των διαφόρων διεργασιών από ένα ΛΣ εξαρτάται από τον τύπο του ΛΣ, αλλά επηρεάζεται και από τις απαιτήσεις των διεργασιών.

Ελαφρές διεργασίες

Κάθε ΛΣ αποτελείται από πολλά επί μέρους προγράμματα, τα οποία εκτελούνται παράλληλα για να εξυπηρετούν τους χρήστες, είναι δηλαδή και αυτό χωρισμένο σε διεργασίες. Το ίδιο μπορεί να γίνει και με τα προγράμματα των χρηστών, να διαιρεθούν δηλαδή σε τμήματα τα οποία εκτελούνται παράλληλα.

> Ο ζαχαροπλάστης (ΚΜΕ) αφιερώνει 5' κάθε φορά στην εκτέλεση μιας συνταγής. Αυτά τα 5' μπορεί να τα μοιράσει σε 5 τμήματα του 1' το καθένα. Σε κάθε τέτοιο τμήμα του 1' μπορεί να ασχοληθεί σε κάθε τμήμα με ένα διαφορετικό στοιχείο του γλυκού το οποίο είναι ανεξάρτητο από τα άλλα. Έτσι μπορεί π.χ. να παρακολουθεί το ψήσιμο της ζύμης στο φούρνο και να ανακατεύει την κρέμα που βράζει.

Όταν όμως τα τμήματα αυτά πρέπει να μοιράζονται διάφορα στοιχεία του προγράμματος, όπως π.χ. μεταβλητές, μπορεί να χρησιμοποιηθεί και ένα εναλλακτικό είδος διεργασιών, οι *ελαφρές διεργασίες* (lightweight processes) ή *νήματα εκτέλεσης* (threads of execution) ή απλούστερα *νήματα* (threads).

Τα νήματα έχουν πολλές ομοιότητες με τις διεργασίες και μια βασική διαφορά: χρησιμοποιούν από κοινού ένα τμήμα μνήμης, στο οποίο έχουν όλα πρόσβαση.

Όπως για κάθε διεργασία, έτσι και για κάθε νήμα το ΛΣ κρατά ένα σύνολο από πληροφορίες σε μια ειδική περιοχή της μνήμης, η οποία αποκαλείται *Σύνολο Ελέγχου Νήματος* (Thread Control Block) ή αλλιώς *ΣΕΝ* (TCB). Το ΣΕΝ κρατά τις απαραίτητες πληροφορίες για κάθε νήμα, αλλά όσες πληροφορίες είναι κοινές μεταξύ τους (π.χ. αυτές για τη μοιραζόμενη μνήμη) καταγράφονται μια φορά μόνο.

> Ο ζαχαροπλάστης πρέπει να θυμάται ή να καταγράφει την πρόοδο για κάθε στάδιο μιας συνταγής που εκτελείται παράλληλα με άλλα στάδια της εκτέλεσής της. Τα σκεύη όμως και τα υλικά που αφορούν τα διάφορα στάδια τα τοποθετεί στο ίδιο μέρος, εκείνο που έχει καθορίσει για τη συγκεκριμένη συνταγή.

Όταν ένα πρόγραμμα διαιρεθεί σε νήματα που εκτελούνται «παράλληλα» (όπως οι διεργασίες), αντί σε ανεξάρτητες διεργασίες, προκαλείται μικρότερη επιβάρυνση στο υπολογιστικό σύστημα, γιατί η μεταγωγή περιβάλλοντος μεταξύ νημάτων είναι πιο γρήγορη από ό,τι μεταξύ διεργασιών. Επιπλέον δεν είναι απαραίτητο πάντα να χρησιμοποιηθούν μηχανισμοί επικοινωνίας από τα νήματα, αφού έχουν για το σκοπό αυτό τη μνήμη που μοιράζονται, κάτι που θα γίνει κατανοητό παρακάτω.

Ένα παράδειγμα προγράμματος που μπορεί να χωριστεί σε νήματα, φαίνεται στο σχήμα. Το πρόγραμμα αυτό υπολογίζει για τις μεταβλητές x και y:

- Με τη ρουτίνα Athroisma, το άθροισμα τους και το αποθηκεύει στη μεταβλητή sum.
- Με τη ρουτίνα Diafora, τη διαφορά τους και την αποθηκεύει στη μεταβλητή diff.
- Με τη ρουτίνα Phliko, το πηλίκο τους και το αποθηκεύει στη μεταβλητή quot.

Οι τρεις αυτοί υπολογισμοί προσδιορίζουν την τιμή διαφορετικών μεταβλητών και ο ένας δεν εξαρτάται από τον άλλο, έτσι μπορούν να γίνουν παράλληλα.

Αν επιλέξουμε να διαιρέσουμε το πρόγραμμα σε τρεις ταυτόχρονες διεργασίες, και οι τρεις χρειάζονται τις μεταβλητές x και y για να κάνουν τους υπολογισμούς τους. Έτσι η λύση αυτή έχει δυο μειονεκτήματα:

→ Η μνήμη σπαταλάται, γιατί έχουμε τρία διαφορετικά αντίγραφα κάθε μεταβλητής στη μνήμη. Εδώ βέβαια πρόκειται για δυο μεταβλητές μόνο, αλλά αν επρόκειτο για πολλά δεδομένα, η σπατάλη θα ήταν πολύ ουσιαστική.

→ Πρέπει να χρησιμοποιηθεί κάποιος μηχανισμός επικοινωνίας που θα εξασφαλίζει ότι οι μεταβλητές x και y των τριών διεργασιών περιέχουν τα ίδια δεδομένα. Αν π.χ. τα δεδομένα δίνονται από το χρήστη, πρέπει η μια από τις τρεις διεργασίες να τα διαβάσει και να τα στείλει στις άλλες δυο, ή μια άλλη διεργασία να τα ζητήσει από το χρήστη και να τα γράψει σε ένα αρχείο, από όπου θα τα διαβάσουν οι υπόλοιπες. Αυτές οι διαδικασίες και προγραμματιστικό κόπο απαιτούν, και επιβαρύνουν το χρόνο εκτέλεσης των διεργασιών, και απαιτούν επιπλέον πόρους του συστήματος (μηχανισμούς επικοινωνίας ή αρχεία).

Η εναλλακτική λύση που δεν έχει τα παραπάνω μειονεκτήματα είναι να χωρίσουμε το πρόγραμμα σε τρία *νήματα*. Τα νήματα αυτά θα μοιράζονται τη μνήμη που καταλαμβάνουν οι δυο μεταβλητές, και θα τη διαβάζουν για να κάνουν τους υπολογισμούς τους. Μόνο ένα αντίγραφο των μεταβλητών θα κρατείται στη μνήμη -έτσι δε γίνεται σπατάλη- και βέβαια δεν τίθεται ζήτημα επικοινωνίας για ανταλλαγή δεδομένων.

```
var
x, y: integer;
sum, diff: integer;
quot: real;

procedure Athroisma;
begin
   sum := x + y
end;

procedure Diafora;
begin
   diff := x - y
end;

procedure Phliko;
begin
   quot := x / y
end;
```

Τα προγράμματα που μπορούν να διαιρεθούν σε τμήματα, κάθε ένα από τα οποία ανατίθεται σε ένα νήμα, ονομάζονται *ταυτόχρονα* (concurrent programs). Ένα τέτοιο πρόγραμμα είναι το προηγούμενο, ή π.χ. το σύστημα κράτησης θέσεων μιας αεροπορικής εταιρίας, όπου κάθε πράκτορας αντιστοιχεί σε ένα νήμα.

Στα επόμενα μαθήματα, ο όρος «Διεργασία» θα χρησιμοποιείται τόσο για τις διεργασίες (που δε χρησιμοποιούν μνήμη από κοινού) όσο και για τα νήματα (τα οποία χρησιμοποιούν μνήμη από κοινού), εκτός και αν αναφέρεται διαφορετικά.

Ανακεφαλαίωση

Ο καταμερισμός του χρόνου της ΚΜΕ μεταξύ πολλών προγραμμάτων δίνει τη δυνατότητα να εκτελούνται αυτά ταυτόχρονα βελτιώνοντας συνολικά τις επιδόσεις του υπολογιστικού συστήματος. Τα προγράμματα που βρίσκονται σε κάποιο στάδιο της εκτέλεσής τους από την ΚΜΕ ονομάζονται διεργασίες. Η διαδικασία εναλλαγής από τη μια διεργασία στην άλλη, δηλαδή η μεταγωγή περιβάλλοντος, απαιτεί την καταγραφή πληροφοριών για τη διεργασία που διακόπτεται ώστε να μπορεί να συνεχιστεί αργότερα σωστά η εκτέλεσή της.

Μια εναλλακτική μορφή οργάνωσης ταυτόχρονων προγραμμάτων, τα οποία διαιρούνται σε τμήματα που μπορούν να εκτελούνται παράλληλα, είναι τα νήματα, μια «ελαφριά» μορφή διεργασιών. Τα νήματα χρησιμοποιούν κοινό τμήμα μνήμης και έτσι δεν απαιτούν μηχανισμούς επικοινωνίας πολλές φορές· επίσης επιβαρύνουν λιγότερο το υπολογιστικό σύστημα.

Γλωσσάριο όρων

Διεργασία	*Process*
Ελαφρή διεργασία	*Lightweight Process*
Κβάντο Χρόνου	*Time Quantum πληθ.-a*
Μεταγωγή Περιβάλλοντος	*Context Switching*
Νήμα	*Thread*
Νήμα Εκτέλεσης	*Thread of Execution*
Σύνολο Ελέγχου Διεργασίας - ΣΕΔ	*Process Control Block - PCB*
Σύνολο Ελέγχου Νήματος - ΣΕΝ	*Thread Control Block - TCB*
Ταυτόχρονο Πρόγραμμα	*Concurrent Program*

Ερωτήσεις

? Ποια είναι τα οφέλη και οι συνέπειες όταν το ΛΣ εναλλάσσει προγράμματα στη χρήση της ΚΜΕ;

? Σε τι διαφέρει η διεργασία από ένα πρόγραμμα; Μπορούν πολλές διεργασίες να αντιστοιχούν στο ίδιο πρόγραμμα;

? Ποια είναι η βασική διαφορά μίας διεργασίας με μία ελαφρή διεργασία;

? Τι είναι η μεταγωγή περιβάλλοντος για μία διεργασία; Είναι η ίδια με τη μεταγωγή περιβάλλοντος για μία ελαφρή διεργασία;

? Πότε συμφέρει να χρησιμοποιήσουμε τις ελαφρές διεργασίες; Πότε νομίζεις ότι πρέπει να τις αποφύγουμε;

Τι μάθαμε σε αυτό το κεφάλαιο

- Το Λειτουργικό Σύστημα είναι ένα σύνολο προγραμμάτων που οργανώνουν και επιβλέπουν τις λειτουργίες του υλικού σε ένα υπολογιστικό σύστημα.
- Τα προγράμματα που βρίσκονται σε κάποιο στάδιο της εκτέλεσής τους από την ΚΜΕ ονομάζονται διεργασίες.
- Οι βασικοί τρόποι απεικόνισης διεργασιών είναι ο γράφος προήγησης και οι εντολές parbegin και parend.
- Το τμήμα μιας διεργασίας που χρησιμοποιεί μοιραζόμενα δεδομένα είναι το κρίσιμο τμήμα της. Μόνο μία διεργασία μπορεί να εκτελεί το κρίσιμο τμήμα της κάθε φορά.
- Οι σηματοφορείς είναι μία πολύ απλή και λειτουργική λύση για το πρόβλημα του κρίσιμου τμήματος.

Βιβλιογραφία – Πηγές

- Παπακωσταντίνου Γ., Ν. Μπιλάλη, Π. Τσανάκα, Λειτουργικά Συστήματα: Αρχές Λειτουργίας, Συμμετρία, 1989.
- Silberschatz A., Peterson J., Galvin P., Operating System Concepts, Addison - Wesley, 1991.