Λειτουργικά Συστήματα

Θεωρία

1.1 Εισαγωγή στα Συστήματα Υπολογιστών

Τα συστήματα υπολογιστών είναι αυτά που βοηθούν στην επίλυση των προβλημάτων του ανθρώπου. Π.χ. το να συνδεθείς σε μία σελίδα είναι ένα πρόβλημα, το να γράψεις ένα κείμενο στον υπολογιστή είναι ένα πρόβλημα, το να παίξεις ένα παιχνίδι είναι ένα πρόβλημα. Ο άνθρωπος χρησιμοποιεί τον υπολογιστεί για να του λύσει όλα αυτά τα προβλήματα και πολλά ακόμη.

Το ότι χρησιμοποιούμε ένα υπολογιστικό σύστημα σημαίνει ότι χρησιμοποιούμε το hardware του υπολογιστή, το software του υπολογιστή δηλαδή προγράμματα και data, δηλαδή δεδομένα για τον υπολογιστή.

Το βασικό στοιχείο των υπολογιστικών συστημάτων είναι ότι εκτελούνται προγράμματα στην CPU. Η CPU είναι ένα chip. Αυτή ψάχνει τα προγράμματα που βρίσκονται μέσα στην μνήμη και θα εκτελέσει ένα ένα τα προγράμματα αυτά. Όταν λέμε μνήμη εννοούμε την RAM. Υπάρχουν όμως κι άλλου είδους μνήμη στους υπολογιστές όπως π.χ. η μνήμη του δίσκου, η cachel.

Κάθε πρόγραμμα που θέλει να εκτελέσει η CPU πρέπει να βρίσκεται στην μνήμη. Εκείνη, θα πάρει ένα πρόγραμμα, θα αναλύσουμε παρακάτω τον τρόπο, το διαβάζει γραμμή γραμμή και το εκτελεί.

Εκτός από το πρόγραμμα υπάρχουν και τα δεδομένα. Το software χωρίς data είναι άχρηστο. Τα data είναι κάθε οποιαδήποτε πληροφορία που μπορεί να υπάρχει στον υπολογιστή.

Το hardware, software, data χρησιμοποιείται για να λύσουμε τα προβλήματα του ανθρώπου με έναν πιο γρήγορο κι αποδοτικό τρόπο.

Υπάρχουν αρκετές είδους μνήμης στον υπολογιστή επειδή κάθε μνήμη έχει τα πλεονεκτήματα και τα μειονεκτήματά του. Ότι αποθηκεύεις στον σκληρό δίσκο είναι μόνιμα. Αν εκεί βάλεις data αυτά θα μείνουν μόνιμα. Αυτό δεν ισχύει στην RAM, εκεί, μόλις κλείσει το

ρεύμα τα δεδομένα χάνονται. Αυτό είναι το πλεονέκτημα του δίσκου απέναντι στην RAM. Το πλεονέκτημα της RAM είναι ότι είναι πολύ πιο γρήγορη από τον δίσκο έως κι εκατομμύρια φορές πιο γρήγορα. Όσο μεγαλύτερη RAM έχει τόσο πιο γρήγορα λειτουργεί ο υπολογιστής.

1.2 Πως δουλεύει το σύστημα ενός υπολογιστή

Ένα πρόγραμμα που γράφεται από τον άνθρωπο είναι ένα πρόγραμμα υψηλού επιπέδου. Π.χ. ένα πρόγραμμα που είναι γραμμένο σε C ή Java ή Python κ.τ.λ. είναι ένα πρόγραμμα υψηλού επιπέδου.

Ο Compiler είναι κι αυτός ένα πρόγραμμα που σκοπός του είναι η μετατροπή ενός προγράμματος υψηλού επιπέδου σε γλώσσα μηχανής, δηλαδή σε μία αλληλουχία από 0 και 1.

Όταν θα γίνει αυτό, το πρόγραμμα αποθηκεύεται μέσα στον σκληρό δίσκου του υπολογιστή. Αυτό σημαίνει ότι αποθηκεύεται μόνιμα εκεί και μόνο εάν το πειράξουμε θα αλλάξει.

Η CPU τώρα δεν μπορεί να έχει πρόσβαση στον σκληρό δίσκο. Εκείνη, έχει πρόσβαση στην RAM ενός υπολογιστή. Η μνήμη RAM είναι πολύ μικρότερη από την μνήμη του σκληρού δίσκου γιατί είναι πολύ πιο γρήγορη κι άρα έχει πολύ μεγαλύτερο κόστος από τον σκληρό δίσκο. Έτσι, ο σκληρός δίσκος που είναι αρκετά φθηνός σε σχέση με την RAM έχει τεράστιο χώρο ενώ η RAM μικρό. Έτσι, δεν γίνεται όλα τα προγράμματα να χωρέσουν μέσα στην RAM αλλά πρέπει να μεταφερθούν τα εκάστοτε προγράμματα που θα χρησιμοποιηθούν από τον σκληρό δίσκο στην RAM και μετά να μεταφερθούν στην CPU και να εκτελεστούν, ένα την φορά για 1 CPU.

Ας πούμε τώρα ότι η CPU θέλει να εκτελέσει ένα πρόγραμμα το οποίο βρίσκεται μέσα στην RAM. Τότε, το πρόγραμμα θα πάρει το πρόγραμμα από την RAM και θα το εκτελέσει. Αν δεν το βρει μέσα στην RAM τότε το πρόγραμμα θα βρίσκεται στον σκληρό δίσκο. Έτσι ο υπολογιστή θα φτιάξει ένα αντίγραφο του προγράμματος και θα μεταφέρει στην RAM και μετά θα το βρει η CPU στην RAM και θα το εκτελέσει.

1.3 Η ανάγκη για λειτουργικό σύστημα

Ένα εύλογο ερώτημα που δημιουργείται από τα προηγούμενα είναι για το πως ο υπολογιστής θα διαλέξει ποια προγράμματα θα μεταφερθούν από τον σκληρό δίσκο στην RAM; Όπως έχουμε πει, σίγουρα όλα τα προγράμματα δεν μπορούν να μεταφερθούν στην RAM γιατί εκείνη είναι πολύ μικρότερη σε χωρητικότητα από τον δίσκο. Μία απάντηση σε αυτό, είναι, ο υπολογιστής να επιλέγει τα προγράμματα που θα χρειαστεί η CPU να εκτελέσει αργότερα, και να τα φέρνει στην RAM. Πως όμως ο υπολογιστής θα το ξέρει αυτό;

Η απάντηση σε αυτό το ερώτημα είναι ότι ο υπολογιστής θα έχει ένα λειτουργικό πρόγραμμα το οποίο θα ασχολείται με αυτό. Συνήθως αυτό το πρόγραμμα είναι γραμμένο σε C και ουσιαστικά ελέγχει τα προγράμματα που θα μεταφερθούν από τον σκληρό δίσκο στην μνήμη καθώς και άλλους πόρους του υπολογιστικού συστήματος.

Ο μακροπρόθεσμος χρόνοπρογραμματιστής του λειτουργικού συστήματος είναι υπεύθυνος για την μεταφορά των προγραμμάτων από τον σκληρό δίσκο στην RAM. Ο μακροπρόθεσμος προγραμματιστής έχει συναρτήσεις οι οποίες θα αποφασίσουν ποιο πρόγραμμα πρέπει να μεταφερθεί.

Ας θεωρήσουμε πως ο μακροπρόθεσμος χρόνοπρογραμματιστής έχει μεταφέρει προγράμματα στην μνήμη. Με τι σειρά θα εκτελεστούν τα προγράμματα στην RAM; Θα εκτελεστούν ανάλογα με έναν αλγόριθμο που είναι ενσωματωμένος στο λειτουργικό σύστημα του υπολογιστή. Με βάση κάποια κριτήρια θα επιλέγει τα προγράμματα που θα πρέπει να μεταφερθούν στην CPU.

<u>Παράδειγμα 1</u>

Εκτελείται το πρόγραμμα το οποίο καταλαμβάνει το μικρότερο μέγεθος στην RAM.

<u>Παράδειγμα 2</u>

Εκτελείται το πρόγραμμα το οποίο μπαίνει πρώτο από τον σκληρό δίσκο στην RAM.

<u>Παράδειγμα 3:</u>

Κάθε δύο ms εκτελείται το κάθε πρόγραμμα ή άμα είναι μεγάλο, εκτελείται ένα μέρος ενός προγράμματος στην CPU και όταν εκτελεστούν μέρη από όλα τα προγράμματα επαναλαμβάνεται η διαδικασία μέχρι να εκτελεστούν όλα.

Η συνάρτηση ενός λειτουργικού συστήματος που επιλέγει ποιο πρόγραμμα θα μεταφερθεί κάθε φορά πρώτο και θα εκτελεστεί από την CPU, καλείται βραχυπρόθεσμος χρονοπρογραμματιστής.

1.4 Είδη λειτουργικών Συστημάτων

Οι τύποι των λειτουργικών συστημάτων είναι 3:

1) Λειτουργικό Σύστημα Batch
2) Πολυπρογραμματιστικό Λειτουργικό Σύστημα
3) Πολυδιεργασιακό Λειτουργικό Σύστημα

1) Το λειτουργικό σύστημα Batch δεν χρησιμοποιείται πλέον. Με βάση αυτό το λειτουργικό σύστημα, μία διεργασία βρίσκεται στην RAM και περιμένει να πάει στην CPU. Καμία όμως άλλη διεργασία δεν μπορεί να υπάρχει στην RAM. Μόλις η CPU δεχθεί την διαδικασία, μία άλλη διαδικασία, μόνη της, μπαίνει στην RAM. Το πρόβλημα με αυτού του είδους το λειτουργικό είναι, ότι όταν μία διεργασία ζητήσει να πάρει κάποιο σήμα I/O (input/output) π.χ. όταν θέλει να πάρει ένα σήμα ότι πατήθηκε ένα πλήκτρο του πληκτρολογίου, τότε η CPU περιμένει έτσι ώστε η διεργασία να πάρει το σήμα και μετά συνεχίζει. Έτσι, χάνεται πολύτιμός χρόνος. Θα μπορούσε όσο η διεργασία ζητάει ένα σήμα I/O η CPU να ασχολείται με άλλη διεργασία.
2) Στο πολυπρογαμματιστικό λειτουργικό σύστημα, πολλές διεργασίες βρίσκονται στην RAM. Η CPU μπορεί να χειρίζεται πάλι μόνο μία διεργασία όμως μπορεί να εναλλάσσει τις διεργασίες έτσι ώστε όταν κάποια ζητήσει I/O να υπάρχει κάποια άλλη που θα μπει στην θέση της μέχρι να πάρει η πρώτη το σήμα.
3) Στο πολυδιεργασιακό λειτουργικό σύστημα έχουμε παραπάνω από μία CPU που κάθεμία δέχεται μία διαφορετική διεργασία και την εκτελεί. Επίσης, πολλές διεργασίες μπορούν να βρίσκονται

στην μνήμη και να περιμένουν να εκτελεστούν από κάποια CPU. Τα σύγχρονα λειτουργικά συστήματα είναι όλα τέτοιου τύπου.

2.1 Διεργασίες

Διεργασία αποτελεί ένα πρόγραμμα το οποίο μπορεί να εκτελεστεί από την CPU. Μία διεργασία αποτελείται από τα εξής μέρη:

1) Κώδικας Προγράμματος: Αυτό το κομμάτι της διεργασίας περιλαμβάνει τον κώδικα που έχει γράψει κάποιος προγραμματιστής για την λειτουργίας της.

2) Τμήμα Global Δεδομένων: Σε αυτό το κομμάτι της διεργασίας αποθηκεύονται οι global μεταβλητές του προγράμματος. Global μεταβλητές είναι οι μεταβλητές ενός προγράμματος που δεν έχουν δημιουργηθεί μέσα σε μία συνάρτηση, δεν είναι δηλαδή local.

3) Τμήμα Static Δεδομένων

4) Στοίβα: Η στοίβα περιλαμβάνει τις συναρτήσεις που κλήθηκαν και υπήρχαν άλλη ή άλλες συναρτήσεις μέσα τους. Η τελευταία συνάρτηση που κλήθηκε βρίσκεται στην κορυφή της στοίβας και η προτελευταία βρίσκεται στην δεύτερη θέση της στοίβας κ.τ.λ.

Για να γίνει κατανοητό αυτό δείτε το εξής παράδειγμα:

$$
\begin{aligned}
&\mathrm{f}_1(x)\{\\
&\quad \ldots\\
&\qquad \mathrm{f}_2(x)\\
&\quad \ldots\\
&\quad \}
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{f}_2(x)\{\\
&\quad \ldots\\
&\qquad \mathrm{f}_3(x)\\
&\quad \ldots\\
&\quad \}
\end{aligned}
$$

$f_3(x)\{$

...

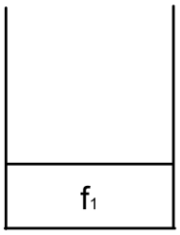
$f_4(x)$

...

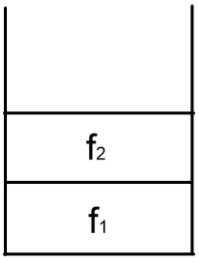
$\}$

Έστω ότι ένα πρόγραμμα έχει τις παραπάνω τέσσερις συναρτήσεις. Η εκτέλεση της διεργασίας ξεκινά με την πρώτη συνάρτηση. Μετά μέσα στην πρώτη συνάρτηση καλείται η δεύτερη, στην δεύτερη συνάρτηση καλείται η Τρίτη συνάρτηση και στην Τρίτη συνάρτηση καλείται η τέταρτη συνάρτηση. Αυτές θα μπούνε στην στοίβα ως εξής:

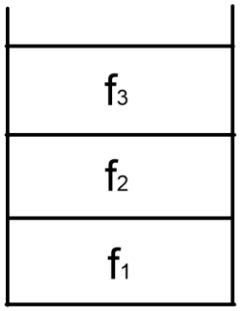
Πρώτα καλείται η πρώτη συνάρτηση άρα μπαίνει στο κάτω μέρος της στοίβας.



Μετά, μέσα στην πρώτη συνάρτηση καλείται η δεύτερη. Έτσι η δεύτερη μπαίνει στην στοίβα από πάνω από την πρώτη. Άρα έχουμε:



Στη συνέχεια, μέσα στην δεύτερη συνάρτηση υπάρχει μία Τρίτη. Όταν καλείται μπαίνει στην κορυφή της στοίβας.



Τώρα, ας υποθέσουμε ότι η Τρίτη συνάρτηση δεν καλεί άλλη συνάρτηση αλλά έχει κάποιον κώδικα που πρέπει να εκτελεστεί. Αυτός ο κώδικας εκτελείται και μετά φθάνουμε στο τέλος της τρίτης συνάρτησης. Έτσι η Τρίτη συνάρτηση ολοκληρώνεται και φεύγει από την κορυφή της στοίβας. Τώρα, στην κορυφή της στοίβας βρίσκεται η δεύτερη συνάρτηση. Εκτελούνται οι εντολές και μετά αφαιρείται και αυτή από το πάνω μέρος της στοίβας. Στο τέλος, εκτελείται η πρώτη συνάρτηση και όταν φθάσουμε στο τέλος της ολοκληρώνεται η διαδικασία.

2.2 Κατάσταση Διεργασίας

Μία διεργασία που εκτελείται αλλάζει κατάσταση. Αυτή η αλλαγή οφείλεται στην τρέχουσα δραστηριότητά της. Οι καταστάσεις στις οποίες μπορεί να βρίσκεται μία διεργασία είναι οι παρακάτω 5:

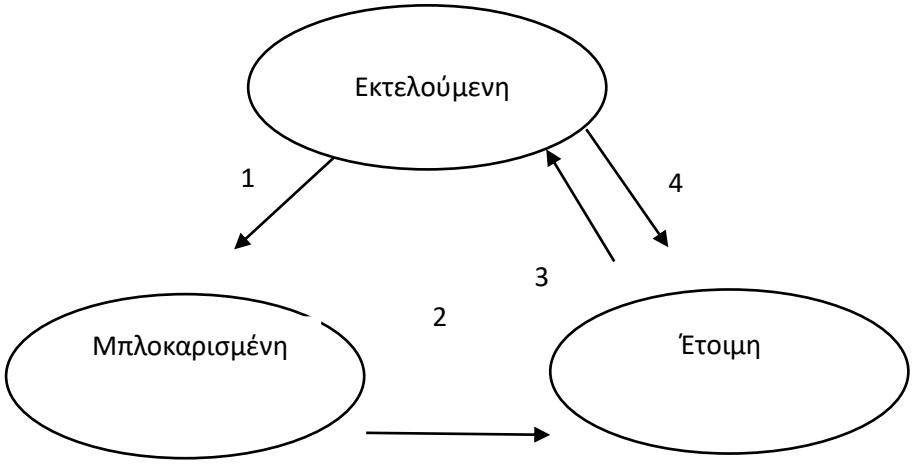
1) Νέα (Μία διεργασία δημιουργείται)
2) Εκτελούμενη (Εκτελούνται οι εντολές της διεργασίας)
3) Αναμονή(Γίνεται παύση της διεργασίας καθώς ο υπολογιστής περιμένει να συμβεί ένα γεγονός όπως π.χ. η λήψη ενός σήματος)
4) Έτοιμη (Η διεργασία είναι έτοιμη να πάει στον επεξεργαστή και περιμένει)
5) Τερματισμός (Ολοκληρώνεται η εκτέλεση της διεργασίας)

Τα ονόματα αυτά που έδωσαν θα μπορούσαν να έχουν ένα συναφές όνομα, δεν είναι αυστηρά καθορισμένα ονόματα. Έχουν διαφορές ανάλογα το λειτουργικό.

Σε έναν επεξεργαστή μπορεί μόνο μία διεργασία να είναι παρούσα. Αν εκτελεστεί, τερματίζεται και μπαίνει μία άλλη. Αν χρειαστεί βγαίνει από τον επεξεργαστή, μπαίνει στην RAM κι έρχεται άλλη διεργασία ανάλογα με τον αλγόριθμο χρονοδρομολόγησης που έχουμε επιλέξει.

Οι καταστάσεις Νέο και Τερματισμός συμβαίνουν στην αρχή και στο τέλος μίας διεργασίας. Μία διεργασία μπορεί να μεταβαίνει μεταξύ των τριών ενδιάμεσων καταστάσεων. Οι δυνατές μεταβάσεις είναι 4:

1) Από εκτελούμενη να γίνει μπλοκαρισμένη
2) Από μπλοκαρισμένα να γίνει Έτοιμη
3) Από έτοιμη να γίνει Εκτελούμενη
4) Από Εκτελούμενη να γίνει Έτοιμη



Η μετάβαση 1 συμβαίνει όταν μία διεργασία πρέπει να εκτελέσει μία κλήση συστήματος.

-

2.3 Πίνακας Ελέγχου Διεργασίας

Για κάθε διεργασία, υπάρχει στο λειτουργικό σύστημα ένας πίνακας ελέγχου διεργασίας που λέγεται και PCB (Process Control Block). Τα κομμάτια από τα οποία αποτελείται αυτός ο πίνακας είναι τα εξής:

1) Κατάσταση Διεργασίας: -
2) Μετρητής του προγράμματος: Ο μετρητής δείχνει ποια θα είναι η επόμενη εκτέλεση εντολής
3) Καταχωρητές: Διαφέρουν ανάλογα με το λειτουργικό σύστημα. Όταν συμβεί μία διακοπή πρέπει η διεργασία να μπορεί να συνεχίσει την εκτέλεσή της αμέσως μετά. –
4) Πληροφορίες Χρονοπρογραμματισμού της Κεντρικής Μονάδας Επεξεργασίας: Περιλαμβάνει τις παραμέτρους χρονοπρογραμματισμού, δηλαδή ποια διεργασία έχει προτεραιότητα και άλλες παραμέτρους χρονοπρογραμματισμού.
5) Πληροφορίες διαχείρισης Μνήμης: -
6) Πληροφορίες Απολογισμού
7) Πληροφορίες κατάστασης Ε/Ε: Είναι μία λίστα που περιλαμβάνει την λίστα συσκευών που Ε/Ε που έχουν αποδοθεί σε μία εργασία π.χ. μία λίστα ανοιχτών αρχείων.

2.4 Χρονοπρογραμματισμός Διεργασιών

Οι διεργασίες που βρίσκονται μέσα στον σκληρό δίσκο τοποθετούνται στην RAM πριν να τρέξουν. Μέσα στην RAM κρατούνται σε μία ουρά. Εκεί βρίσκονται σε κατάσταση ετοιμότητας και είναι έτοιμες να εκτελεστούν. Η κεφαλή μίας ουράς περιέχει δείκτες για την πρώτη και την τελευταία διεργασία που βρίσκεται στην ουρά.

Μία συνηθισμένη αναπαράσταση χρονοπρογραμματισμού διεργασιών περιλαμβάνει ένα διάγραμμα ουρών. Υπάρχουν δύο τύποι ουρών:

1) Η ουρά έτοιμων διεργασιών
2) Σύνολο ουρών συσκευών

Μία νέα διεργασία που μπαίνει στην RAM τοποθεταίται στην ουρά έτοιμων διεργασιών και περιμένει για την εκτέλεσή της. Μπορεί να πραγματοποιήσει ένα αίτημα π.χ. το κλικ ενός ποντικιού και να περιμένει στην ουρά Εισόδου Εξόδου για να τρέξει.

2.5 Χρονοδρομολογητές

Το λειτουργικό σύστημα πρέπει να διαλέξει ποια διεργασία θα τρέξει από την ουρά διεργασιών. Υπάρχουν 3 ειδών χρονοδρομολογητών:

1) Μακροπρόθεσμος Χρονοπρογραμματιστής: Επιλέγει ποιες διεργασίες από τον δίσκο θα μεταφερθούν στην RAM.
2) Βραχυπρόθεσμος Χρονοδρομολογητής: Επιλέγει ποιες διεργασίες θα πάνε από την ουρά χρονοπρογραμματισμού στην κεντρική μονάδα επεξεργασίας.
3) Μεσοπρόθεσμος Χρονοδρομολογητής: Μερικές φορές μία διεργασία που βρίσκεται στην RAM χρειάζεται να φύγει από εκεί και να πάει στον δίσκο. Αυτό το αναλαμβάνει ο μεσοπρόθεσμος χρονοδρομολογητής

2.6 Δημιουργία Διεργασίας

Μία διεργασία καθώς εκτελείται μπορεί να δημιουργήσει αρκετές νέες διεργασίες. Η αρχική διεργασία που δημιουργεί καλείται γονική διεργασία και οι νέες διεργασίες που δημιουργεί καλούνται θυγατρικές διεργασίες της διεργασίας αυτής.

Αν μία διεργασία δημιουργήσει νέες διεργασίες τότε μπορεί να συμβούν 2 πράγματα:

1) Η γονική διεργασία τρέχει παράλληλα μαζί με τις θυγατρικές διεργασίες της

2) Η γονική διεργασία περιμένει την άλλη ή τις άλλες θυγατρικές διεργασίες να τερματίσουν την εκτέλεσή τους για να συνεχίσει.

Μία διεργασία τερματίζεται όταν εκτελείται και η τελευταία εντολή της και καταλήγει να ζητήσει από το λειτουργικό σύστημα για να την διαγράψει. Η εντολή που χρησιμοποιείται για να γίνει αυτό είναι η exit().

Ακόμα, υπάρχει περίπτωση μία διεργασία να τελειώσει αν η γονική διεργασία συνήθως, δώσει μία εντολή που να προκαλέσει τον τερματισμό μίας θυγατρικής διεργασίας.

Αυτό μπορεί να συμβεί για πολλούς λόγους:

1) Αν η θυγατρική διεργασία χρησιμοποιήσει πολλούς πόρους.
2) Το έργο που της έχει ανατεθεί δεν είναι πλέον χρήσιμο.
3) Αν τερματιστεί η γονική διεργασία και το λειτουργικό σύστημα δεν επιτρέψει την συνέχιση των θυγατρικών διεργασιών.