ΕΝΟΤΗΤΑ 1

ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ

Περιεχόμενα

1. Τι είναι Λ.Σ. Ρόλος και Στόχοι ενός Λ.Σ.

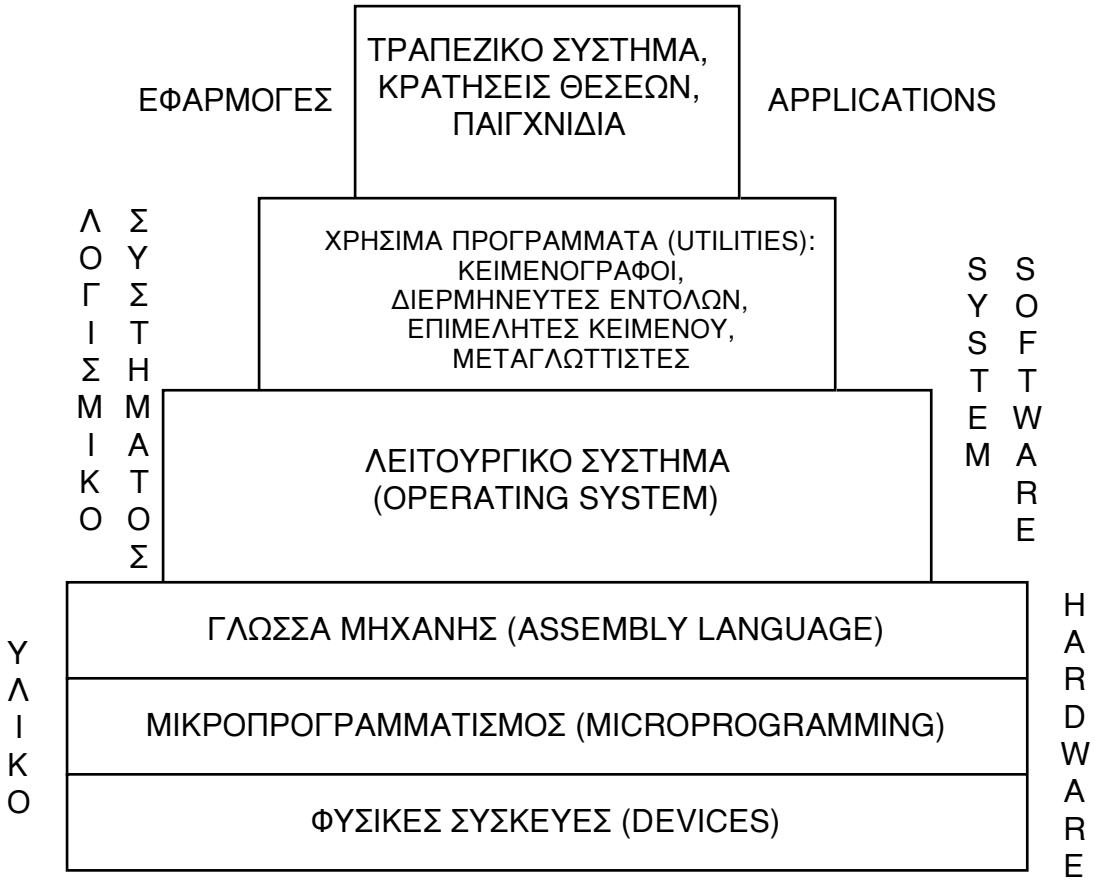
- Λειτουργικό Σύστημα είναι ένα πρόγραμμα που ελέγχει την εκτέλεση των προγραμμάτων εφαρμογών και προσφέρει μία διασύνδεση (interface) μεταξύ του χρήστη και του Η/Υ.

- Στόχοι ενός λειτουργικού συστήματος:
  - Παροχή *διευκολύνσεων* στη χρήση του Η/Υ.
  - Πιο *αποδοτική* χρήση των διαθέσιμων πόρων (resources) του συστήματος.
  - Δυνατότητα για εύκολη *επέκταση* με επιπλέον λειτουργικότητες αλλά και διόρθωση υπάρχοντων λαθών.

- Οι στόχοι αυτοί καθορίζουν και το διπλό ρόλο που παίζει ένα λειτουργικό σύστημα σε σχέση με το χρήστη και τον Η/Υ:
  - Ως μία *εκτεταμένη* (extended) ή *ιδεατή* (virtual) μηχανή που παρέχει μία διασύνδεση μεταξύ χρήστη και Η/Υ.
  - Ως *διαχειριστής* των πόρων του συστήματος (resource manager).

Σημειώστε επίσης ότι η ανάπτυξη των Λ.Σ. οδηγεί αναγκαστικά στη δημιουργία πιο πολύπλοκων σε χρήση και σε λειτουργία περιβαλλόντων:
- Το Multics ανακοινώθηκε το 1963 και έτρεξε το 1969.
- Το OS 360 όταν πρωτοβγήκε στην αγορά είχε 1000 λάθη (bugs).
- Ένα μικρό Λ.Σ. έχει μέγεθος 100K και ένα μεγάλο 10M γραμμών.
- Το Windows NT της Microsoft υπέφερε επί 7 χρόνια με προβλήματα.
- Η δημιουργία ενός Λ.Σ. παίρνει μεταξύ 100-1000 ανθρώπινα χρόνια (man years).

1.1. Ο ρόλος του Λ.Σ. ως εκτεταμένης/ιδεατής μηχανής



- Προσεγγίζοντας το θέμα του ρόλου ενός Λ.Σ. από την κορυφή προς τη βάση διαπιστώνουμε ότι κυρίαρχη φροντίδα του είναι να απομονώσει τη γυμνή μηχανή από τον χρήστη στον οποίο παρουσιάζει μία πιο απλή, εύχρηστη και κατανοητή παραλλαγή της, δηλαδή μία *εκτεταμένη* ή *ιδεατή* μορφή.

1.1. Ο ρόλος του Λ.Σ. ως εκτεταμένης/ιδεατής μηχανής (συνέχεια)

- Η διασύνδεση χρήστη-μηχανής που δημιουργεί το Λ.Σ. έχει ως σκοπό την παροχή των ακόλουθων υπηρεσιών:
  - — Διευκόλυνση στη δημιουργία προγραμμάτων, επιτρέποντας τη χρήση του κατάλληλου λογισμικού του συστήματος (διορθωτές κειμένου, μεταγλωττιστές, απασφαλματωτές, κλπ.). [Η παροχή των υπηρεσιών αυτών δεν είναι κατ' ανάγκη ευθύνη του Λ.Σ.]
  - — Διαχείρηση του συστήματος κατά την εκτέλεση ενός προγράμματος (π.χ. μεταφορά δεδομένων και εντολών προς και από την κύρια μνήμη) και ετοιμασία συσκευών για χρήση.
  - — Εύκολη προσπέλαση στις συσκευές εισόδου/εξόδου μέσω απλών εντολών διαβάσματος και γραψίματος και παροχή αφαιρετικών μηχανισμών από τις ιδιαιτερότητες της κάθε συσκευής.
  - — Ελεγχόμενη πρόσβαση σε αρχεία (είδη αρχείων, ταυτόχρονη προσπέλαση από πολλούς χρήστες, κλπ.).
  - — Ελεγχόμενη πρόσβαση στο σύστημα (ασφάλεια δεδομένων, μη εξουσιοδοτημένη χρήση πληροφοριών, κλπ.).
  - — Αντιμετώπιση λαθών και προσπαθειών για λανθασμένη χρήση του συστήματος (π.χ. διαίρεση με 0, διακοπή λειτουργίας συσκευής).
  - — Στατιστική χρήσης του συστήματος (κατάλληλη για καλύτερη προσαρμογή του συστήματος στις ανάγκες των χρηστών του, υπολογισμό λογαριασμών, κλπ.).

1.2. Ο ρόλος του Λ.Σ. ως διαχειριστή πόρων

- Προσεγγίζοντας το θέμα του ρόλου ενός Λ.Σ. από τη βάση προς την κορυφή διαπιστώνουμε ότι το Λ.Σ. οφείλει να:
  - — ελέγχει ποιος χρησιμοποιεί ποιους πόρους,
  - — διαβαθμίζει τις απαιτήσεις για πόρους και χρεώνει για τη χρήση τους,
  - — επιλαμβάνεται για τις αντικρουόμενες διεκδικήσεις από διαφορετικά προγράμματα και χρήστες.

2. Ιστορία και Εξέλιξη των Λ.Σ.

- Η εξέλιξη των λειτουργικών συστημάτων σχετίζεται σε μεγάλο βαθμό με αυτή των Η/Υ.

- **1η Γενεά (1945-1955):** Λυχνίες Κενού και Κάρτες Καλωδιακών Συνδέσεων.
  - — Σχεδόν ανύπαρκτη η ύπαρξη και χρήση λειτουργικών συστημάτων.
  - — Το "πρόγραμμα" αποτελείτο από κάρτες καλωδιακών συνδέσεων (plugboards) που "έτρεχαν" στον Η/Υ (αν και όταν κάποιες από τις περίπου 20.000 λυχνίες κενού δεν ήταν καμένες).
  - — Προς τις αρχές της δεκαετίας του '50 γίνεται "αναβάθμιση" με τη χρήση διάτρητων καρτών (punched cards) αντί για κάρτες καλωδιακών συνδέσεων.
  - — Ευθύνη του χρήστη να αναλάβει ο ίδιος όλες τις λειτουργίες που σχετίζονται με την εκτέλεση κάποιου προγράμματος: φόρτωμα του μεταγλωττιστή, βιβλιοθηκών, κλ.π , παραγωγή συμβολικής γλώσσας, φόρτωμα του συμβολομεταφραστή και παραγωγή γλώσσας μηχανής.

- **2η Γενεά (1955-1965):** Κρυσταλλοτρίοδοι και Συστήματα Μαζικής Επεξεργασίας.
  - — Η χρήση κρυσταλλοτριόδων (transistors) έχει θετικό αντίκτυπο και στην ανάπτυξη του λογισμικού. Αναπτύσσεται η έννοια της <u>εργασίας</u> (job) αποτελούμενης από ένα ή περισσότερα προγράμματα γραμμένα σε κάποια γλώσσα προγραμματισμού (π.χ. FORTRAN) μαζί με τα χρειαζόμενα δεδομένα. Κάθε εργασία διαβάζονταν μέσα στον Η/Υ με τη χρήση διάτρητων καρτών.
  - — <u>Σειριακή επεξεργασία</u>. Πρώτα ολοκληρώνεται μία εργασία πριν αρχίσει άλλη.
  - — <u>'Ελλειψη χρονοδρομολόγησης</u> (scheduling). Κάποιος χρήστης μπορούσε να "κλείσει" τον Η/Υ για μία ώρα αν και τον χρειαζόταν τελικά μόνο 3 τέταρτα.
  - — <u>Σπατάλη χρόνου</u> στην προετοιμασία (set up) του συστήματος με τη φόρτωση του κατάλληλου λογισμικού συστήματος (μεταφραστές γλωσσών, κλπ.).

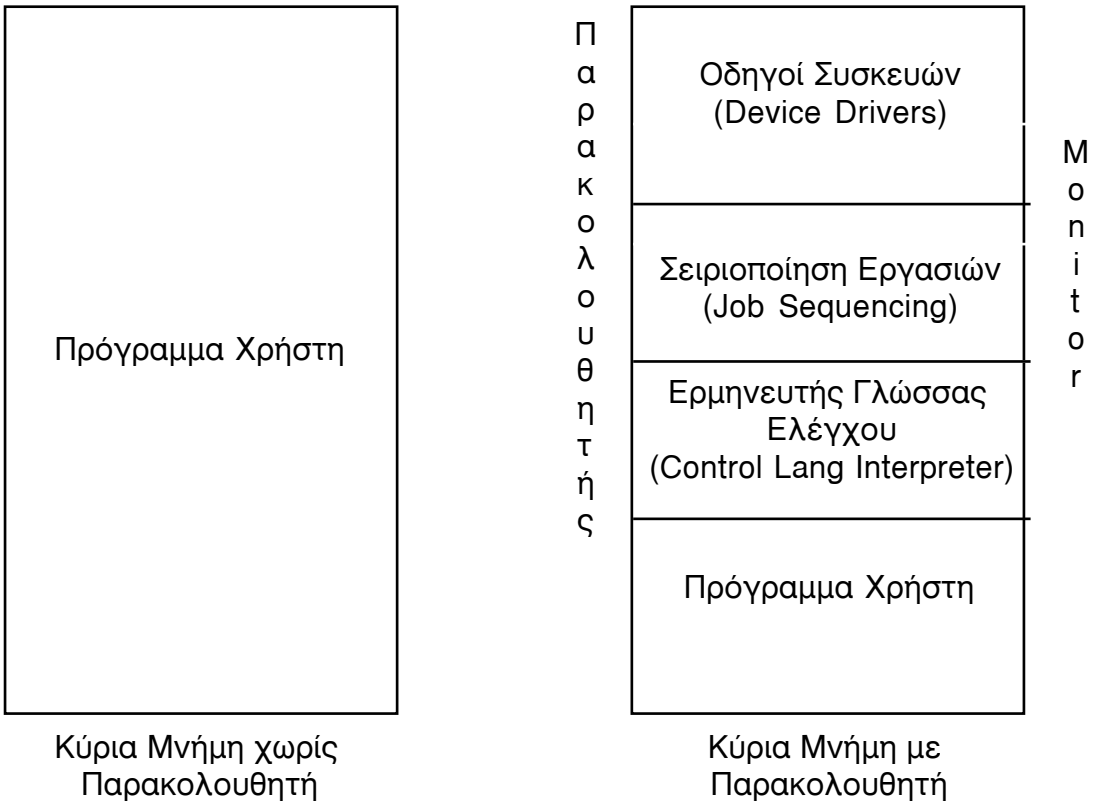
2. Ιστορία και Εξέλιξη των Λ.Σ. (συνέχεια)

— Βελτίωση του προβλήματος με την ανάπτυξη συστημάτων μαζικής σειριακής επεξεργασίας (batch processing). Το σύστημα διαβάζει τις εργασίες όλες μαζί και μόλις τελειώσει μία φορτώνει αμέσως την επόμενη εξοικονομώντας έτσι σημαντικό χρόνο. Υπεύθυνο για τον χειρισμό της όλης διεργασίας είναι ένα πρόγραμμα ονομαζόμενο παρακολουθητής (monitor). Ο παρακολουθητής είναι ακόμα υπεύθυνος για το φόρτωμα του λογισμικού συστήματος (π.χ. μεταφραστής) που χρειάζεται κάθε εργασία.

— Ο παρακολουθητής είναι ένα λειτουργικό σύστημα σε εμβρυϊκή μορφή. Υπάρχει ανάγκη για περισσότερη κύρια μνήμη για να εξυπηρετηθούν οι ανάγκες του.

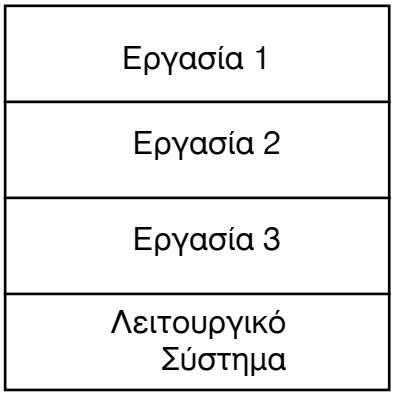
— Αναπτύσσονται στοιχειώδεις γλώσσες ελέγχου εργασιών (job control languages) για την παροχή οδηγιών στον παρακολουθητή.

— Παραμένει το πρόβλημα της σπατάλης χρόνου από τον επεξεργαστή όταν μία εργασία εκτελεί εντολές εισόδου ή εξόδου. Ανάπτυξη τεχνικών όπως *προσωρινή καταχώρηση* δεδομένων εισόδου/εξόδου (buffering) και *ετεροχρονισμός* (spooling - Simultaneous Peripheral Operation On Line).



- **3η Γενεά (1965-1980):** Ολοκληρωμένα Κυκλώματα και Πολυπρογραμματισμός.

— Η εισαγωγή ολοκληρωμένων κυκλωμάτων στην κατασκευή των Η/Υ οδήγησε στη βελτίωσή τους σε ταχύτητα και μνήμη και επέτρεψε την ανάπτυξη του πολυπρογραμματισμού (multiprogramming), δηλαδή της ταυτόχρονης εκτέλεσης περισσοτέρων της μίας εργασιών.



Κύρια Μνήμη

Μόλις μία εργασία σταματούσε να χρησιμοποιεί τον επεξεργαστή για να εκτελέσει μία λειτουργία εισόδου/εξόδου, το λειτουργικό σύστημα ξεκινούσε την εκτέλεση άλλης εργασίας. Η κύρια μνήμη επομένως ήταν μοιρασμένη μεταξύ του λειτουργικού συστήματος και ενός αριθμού εργασιών.

— Επιπλέον, αν περισσότερες από μία εργασίες ήταν υποψήφιες για εκτέλεση, έπρεπε να ορισθούν κάποια κριτήρια με βάση τα οποία κάποια από αυτές θα επιλεγόταν να εκτελεσθεί. Αυτό οδήγησε στην ανάπτυξη της χρονοδρομολόγησης (scheduling).

— Η δυνατότητα αποθήκευσης ενός αριθμού εργασιών στην κύρια μνήμη οδήγησε στην ανάγκη για διαχείρηση της μνήμης (memory management).

— Μία παραλλαγή του πολυπρογραμματισμού που αναπτύχθηκε με σκοπό την όσο το δυνατόν μεγαλύτερη εξοικονόμηση χρόνου ήταν ο καταμερισμός χρόνου (time sharing), όπου ένας αριθμός χρηστών ήταν συνδεδεμένοι ταυτόχρονα με τον Η/Υ με τη χρήση τερματικών και μπορούσαν να διορθώσουν προγράμματα, να εκτελέσουν εργασίες, κλπ. Το πρώτο σύστημα καταμερισμού χρόνου ήταν το CTSS (Compatible Time Sharing System) στο MIT το 1962.

2. Ιστορία και Εξέλιξη των Λ.Σ. (συνέχεια)

- **4η Γενεά (1980-1990):** LSI/VLSI και Προσωπικοί Υπολογιστές.
  - — Με την ανάπτυξη των κυκλωμάτων μεγάλης και πολύ μεγάλης ολοκλήρωσης (LSI, VLSI) μειώθηκε σημαντικά το κόστος κατασκευής των Η/Υ και αναπτύχθηκαν οι προσωπικοί Η/Υ.
  - — Ανάπτυξη λειτουργικών συστημάτων ενός χρήστη (single user) με έμφαση στη φιλικότητα προς τον χρήστη (π.χ. MS-DOS, Windows, OS/2).
  - — Ανάπτυξη ισχυρών σταθμών εργασίας (workstations) και δικτύων (networks) που οδήγησαν στην ανάπτυξη λειτουργικών συστημάτων για δίκτυα (network operating systems) και κατανεμημένων λειτουργικών συστημάτων (distributed operating systems).

3. Βασικές Έννοιες και Επιτεύγματα των Λ.Σ.

- **Διεργασία (process):**
  - — Η οντότητα εκείνη που εκτελείται από τον επεξεργαστή. Ανά πάσα στιγμή στο σύστημα υπάρχουν πολλές διεργασίες, μία από τις οποίες εκτελείται ενώ οι υπόλοιπες περιμένουν τη σειρά τους.
  - — Μία διεργασία μπορεί να δημιουργήσει άλλες θυγατρικές διεργασίες (child processes) και να περιμένει (ή όχι) τον τερματισμό τους πριν συνεχίσει τη δικιά της εκτέλεση. Π.χ. μία διεργασία που αντιστοιχεί στη χρήση του φλοιού (shell) δημιουργεί μία άλλη διεργασία για κάθε απαίτηση ενός χρήστη.
  - — Για να μπορεί να αναστέλλεται προσωρινά η εκτέλεσή της, μία διεργασία παριστάνεται στο σύστημα με ένα σύνολο πληροφοριών όπως οι τιμές των καταχωρητών όταν σταμάτησε η εκτέλεσή της, ο ιδιοκτήτης της, κλπ. Όλες οι σχετικές με κάποια διεργασία πληροφορίες αποθηκεύονται σε ένα πίνακα διεργασιών (process table) αποτελούμενο από μία δομή για κάθε διεργασία, με όλες τις δομές συνδεδεμένες μεταξύ τους υπό μορφή συνδεδεμένης λίστας (linked list).

3. Βασικές Έννοιες και Επιτεύγματα των Λ.Σ. (συνέχεια)

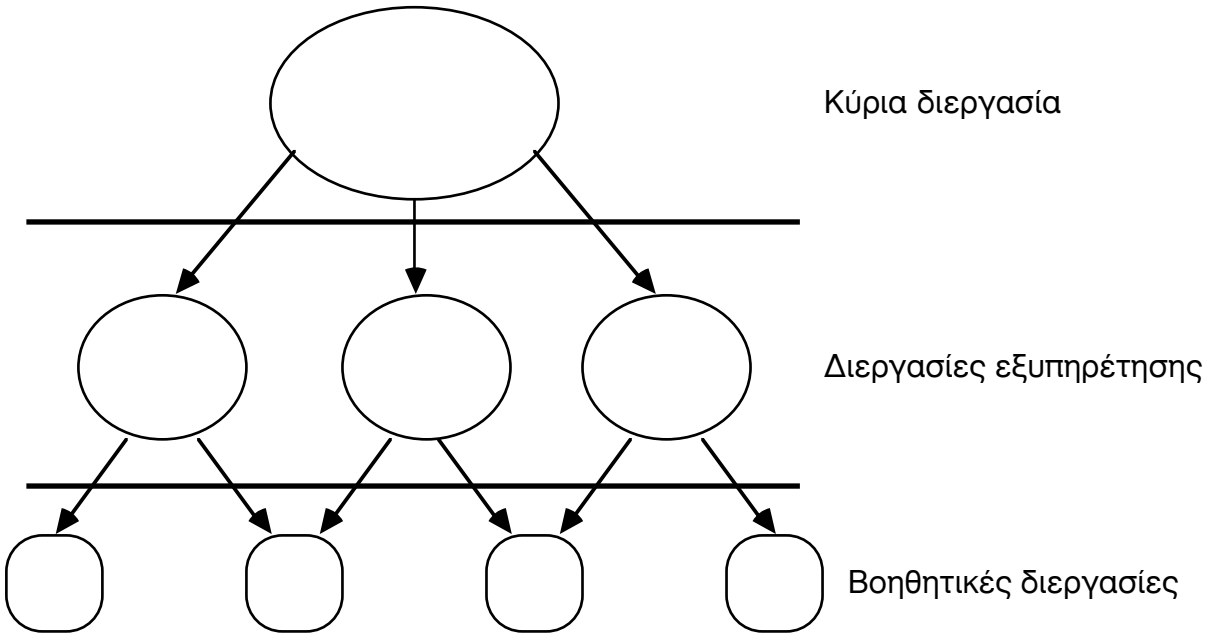
— Δημιουργείται έτσι η ανάγκη για διαχείρηση των διεργασιών (process management) και επίλυση προβλημάτων όπως κακός συγχρονισμός μεταξύ διεργασιών, μη καθοριστική (non deterministic) συμπεριφορά και αδιέξοδα (deadlocks).

- **Διαχείρηση Μνήμης (Memory Management):**
  - — Μοίρασμα της μνήμης μεταξύ των διεργασιών που υπάρχουν στο σύστημα.
  - — Αυτόματη δέσμευση και αποδέσμευση μνήμης.
  - — Μακρόχρονη αποθήκευση πληροφοριών.
  - — Χρήση ιδεατής μνήμης (virtual memory): οι διεργασίες και τα προγράμματα βλέπουν τη μνήμη από μία λογική οπτική γωνία χωρίς να χρειάζεται να γνωρίζουν πόση πραγματική μνήμη είναι στην ουσία διαθέσιμη.
- **Προστασία και Ασφάλεια των Πληροφοριών του Συστήματος:**
  - — Η ταυτόχρονη ύπαρξη πολλών χρηστών αλλά και διεργασιών σε κάποιο σύστημα δημιουργεί ανάγκες για ασφάλεια των πληροφοριών από τυχαία ή εσκεμμένη προσπάθεια αλλοίωσης/καταστροφής τους.
  - — Οι διάφοροι μέθοδοι που έχουν υιοθετηθεί κυμαίνονται από απαγόρευση χρήσης του ίδιου λογισμικού από περισσότερες από μία διεργασίες μέχρι τη χρήση μηχανισμών όπως λίστες ελέγχου προσπέλασης, προσδιοριστές δικαιωμάτων, κλπ.
- **Διαχείρηση των Πόρων (Resource Management) και Χρονοδρομολόγηση (Scheduling):**
  - — Όσο το δυνατόν πιο αποτελεσματική διαχείρηση των πόρων του συστήματος με βάση τα εξής κριτήρια:
    i) *δικαιοσύνη* (fairness), δηλαδή όλες οι διεργασίες κάποια στιγμή να εξυπηρετηθούν,
    ii) αναγνώριση τυχόν *ιδιαιτεροτήτων* κάποιας διεργασίας, όπως σύντομο χρόνο εκτέλεσης ή μικρή απαίτηση σε μνήμη,
    iii) *αποδοτικότητα* (efficiency) του συστήματος, δηλαδή μεγιστοποίηση του αριθμού των εργασιών που ολοκληρώνονται στη χρονική μονάδα μέτρησης (διεκπεραιωτική ικανότητα, throughput), ελαχιστοποίηση του χρόνου απόκρισης (response time) και εξυπηρέτηση όσο το δυνατόν περισσότερων χρηστών.

4. Δομή ενός Λειτουργικού Συστήματος

Η παρουσίαση της δομής ενός λειτουργικού συστήματος “εκ των έσω” οδηγεί στο διαχωρισμό τους σε 2 κατηγορίες:

- **Μονολιθικά συστήματα (monolithic systems).**
  - Το Λ.Σ. αποτελείται από μία συλλογή διεργασιών, κάθε μία των οποίων μπορεί να καλέσει οποιαδήποτε άλλη. Κάθε διεργασία είναι ορατή σε οποιαδήποτε άλλη. Ακόμα και εδώ όμως μπορεί να υπάρχει μία στοιχειώδης ιεραρχία:



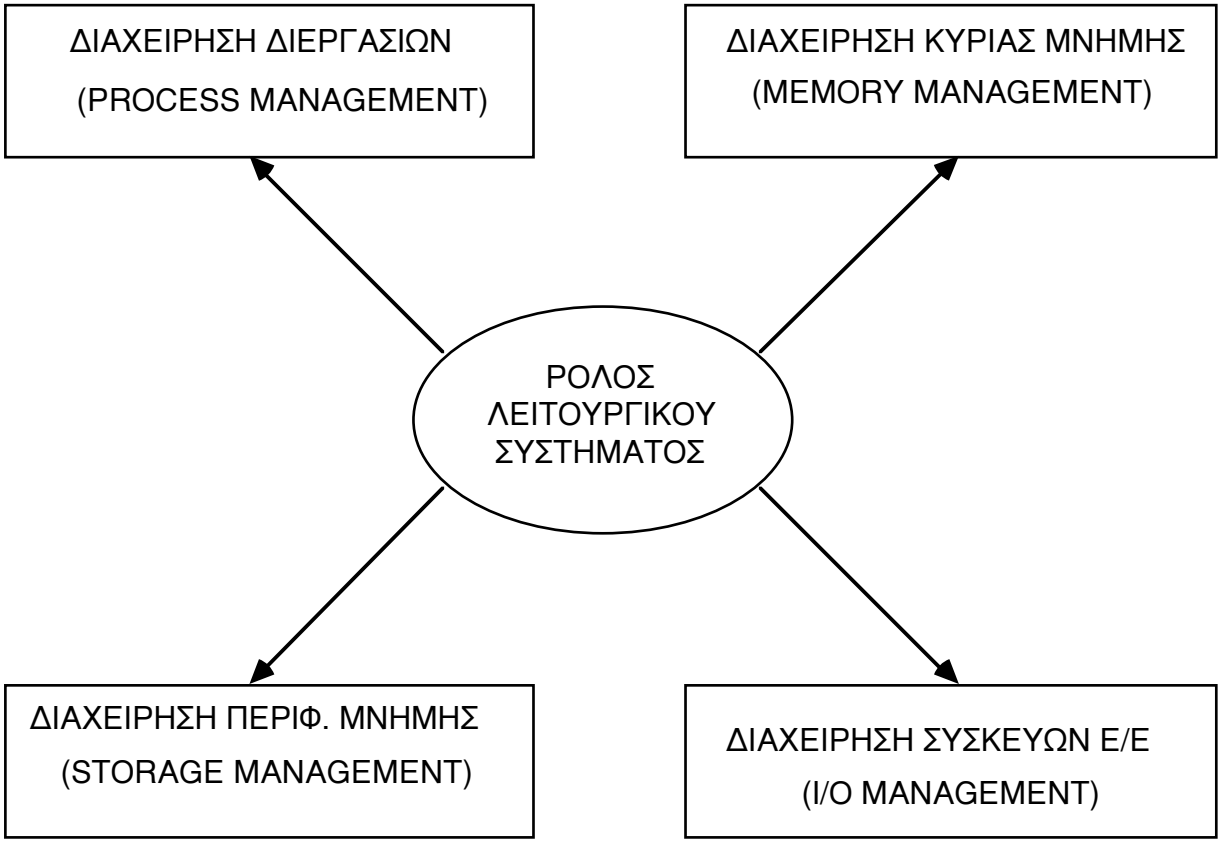
  - Το κυρίως πρόγραμμα επικαλείται την ενεργοποίηση μίας ομάδας διεργασιών εξυπηρέτησης οι οποίες με τη σειρά τους υποβοηθούνται από άλλες βοηθητικές διεργασίες.

4. Δομή ενός Λειτουργικού Συστήματος (συνέχεια)

- **Στρωματοποιημένα Συστήματα (layered systems).**
  - Η γενίκευση της προσέγγισης του προηγούμενου σχήματος οδήγησε στην οργάνωση των λειτουργικών συστημάτων σαν μία ιεραρχία επιπέδων, το καθένα από τα οποία επικοινωνεί κατ' εξοχήν μόνο με το επόμενο και προηγούμενο. Ο αριθμός των επιπέδων σε ένα Λ.Σ. ποικίλλει από το ένα Λ.Σ. στο άλλο. Οι Denning και Brown περιγράφουν μία δομή αποτελούμενη από 13 συνολικά επίπεδα:
    - Επίπεδα 1 έως 4 αποτελούν το υλικό (κυκλώματα, γλώσσα μηχανής) και ουσιαστικά δεν είναι μέρος του Λ.Σ.
    - Επίπεδο 5: Διαχείριση στοιχειωδών διεργασιών (παροχή εντολών όπως suspend, resume, wait, signal) και υλοποίηση μεταγωγής περιβάλλοντος.
    - Επίπεδο 6: Διαχείριση περιφερειακών συσκευών (π.χ. διάβασμα από ή γράψιμο σε δίσκο).
    - Επίπεδο 7: Ιδεατή μνήμη (υποστήριξη λογικών διευθύνσεων μνήμης).
    - Επίπεδο 8: Επικοινωνία μεταξύ διεργασιών (pipes).
    - Επίπεδο 9: Διαχείριση αρχείων (create, open, write) στο επίπεδο του χρήστη (π.χ. μέσα από το ρεπερτόριο εντολών μίας γλώσσας προγραμματισμού).
    - Επίπεδο 10: Διαχείριση περιφερειακών συσκευών και παροχή ομοιογενών μηχανισμών πρόσβασης του χρήστη σε όλα τα είδη τους, έστω και αν αυτά παρουσιάζουν μεταξύ τους σημαντικές διαφορές στη λειτουργία τους.
    - Επίπεδο 11: Κατάλογοι.
    - Επίπεδο 12: Διαχείριση διεργασιών σε επίπεδο χρήστη.
    - Επίπεδο 13: Προγραμματισμός συστήματος με κατάλληλες γλώσσες (όπως το κέλυφος του Unix).
  - Σημειωτέον ότι τα επίπεδα 5 έως 8 εστιάζουν τη λειτουργία τους σε διαδικασίες που επιτελούνται αποκλειστικά στο περιβάλλον ενός επεξεργαστή ενώ στα επίπεδα 8 έως 13 δύναται να γίνεται αναφορά και σε εξωτερικά του άμεσου περιβάλλοντος ενός μεμονωμένου Η/Υ αντικείμενα με πρόσβαση σε άλλους Η/Υ μέσω χρήσης δικτύων.

4. Δομή ενός Λειτουργικού Συστήματος (συνέχεια)

- Επιστρέφοντας στον ρόλο του λειτουργικού συστήματος βλέπουμε ότι εστιάζεται σε 4 βασικές περιοχές:



5. Μερικά τυπικά Λειτουργικά Συστήματα

- Συστήματα ενός χρήστη: MS-DOS, OS/2, Apple Macintosh, Windows NT, Windows 2000.
- Συστήματα πολλών χρηστών: Unix, MVS, VM.