Λειτουργικά Συστήματα (HY321)

Διάλεξη 1:
Εισαγωγή

Χρήστος Δ. Αντωνόπουλος
2/3/2017

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Εσείς;

- Έτος;
- Υποχρεωτικό / Επιλογής;
- Background;

- Τι περιμένετε από το μάθημα;

Περιεχόμενο Μαθήματος

- Εισαγωγή, γενικά περί λειτουργικών συστημάτων
  - Γενικά στοιχεία αρχιτεκτονικών
- Νήματα & Διεργασίες
- Σύντομη επισκόπηση του συγχρονισμού, αδιεξόδων
- Χρονοδρομολόγηση
- Διαχείριση μνήμης
  - Ιδεατή μνήμη
  - Σελιδοποίηση
  - Διαχείριση σε επίπεδο χρήστη
- Δίσκοι
  - Αρχεία, κατάλογοι, οργάνωση
  - Δρομολόγηση δίσκων
  - Caching
  - Solid State Disks (SSDs)
- I/O
  - Interrupts DMA
- Πολιτικές και μηχανισμοί προστασίας
- Ζητήματα Ασφάλειας
- Virtualization
- Διαχείριση κατανάλωσης ισχύος

Βιβλιογραφία

- Συγγράμματα:
  - «Λειτουργικά Συστήματα», Silberschatz, Galvin, Gagne, Εκδόσεις Ίων.
  - «Λειτουργικά Συστήματα. Αρχές Σχεδίασης», William Stallings, Εκδόσεις Τζιόλα.
  - «Σύγχρονα Λειτουργικά Συστήματα», Andrew Tanenbaum, Εκδόσεις Κλειδάριθμος.

- Προτεινόμενα άλλα συγγράμματα:
  - «Operating Systems Design and Implementation», Andrew Tanenbaum, Prentice Hall.
  - «Linux Kernel Development»,Robert Love, εκδόσεις Novell Press.

Απαιτήσεις - Αξιολόγηση

- Τελική εξέταση
- Ομαδική εργασία (προαιρετική για όσους παίρνουν το μάθημα ως υποχρεωτικό, υποχρωτική για όσους παίρνουν το μάθημα ως επιλογής)
  - Ομάδες 3 ατόμων
  - 4 φάσεις (1 εισαγωγική, 3 κανονικές)
  - Λειτουργικό σύστημα Linux
- Αξιολόγηση με εργασία:
  - Πρέπει εξέταση >= 5 και εργασία >=5
  - Τελικός βαθμός = 0,6 * εξέταση + 0,4 * εργασία
  - Δέσμευση με την παράδοση της 2ης εργασίας
  - Ισχύουν οι εργασίες ακαδημαϊκού έτους 2014-15
- Αξιολόγηση χωρίς εργασία
  - Πρέπει εξέταση >= 5
  - Τελικός βαθμός = 5 + (εξέταση – 5) * 0,4
  - Μέγιστος τελικός βαθμός: 7

Λοιπές Πληροφορίες

- Ιστοσελίδα μαθήματος:
  http://www.e-ce.uth.gr/courses/CE321
- Λίστα ηλεκτρονικού ταχυδρομείου:
  http://support.inf.uth.gr/mailman/listinfo/ce321
- Forum μαθήματος:
  http://courses.e-ce.uth.gr/codingforums

Λοιπές Πληροφορίες

- Επικοινωνία με τον διδάσκοντα
  - Γραφείο: Γκλαβάνη, Β3/5
  - Ώρες γραφείου: Δείτε σελίδα, κλείστε ραντεβού από το http://cdantonop.youcanbook.me
  - E-mail cda@e-ce.uth.gr cdantonop@gmail.com
- Επικοινωνία με τους μεταπτυχιακούς βοηθούς του μαθήματος:
  - Παναγιώτης Κουτσοβασίλης

    Γραφείο: Γκλαβάνη, Β4/8

    E-mail: pkoutsovasilis@e-ce.uth.gr
  - Βασίλης Βασιλειάδης

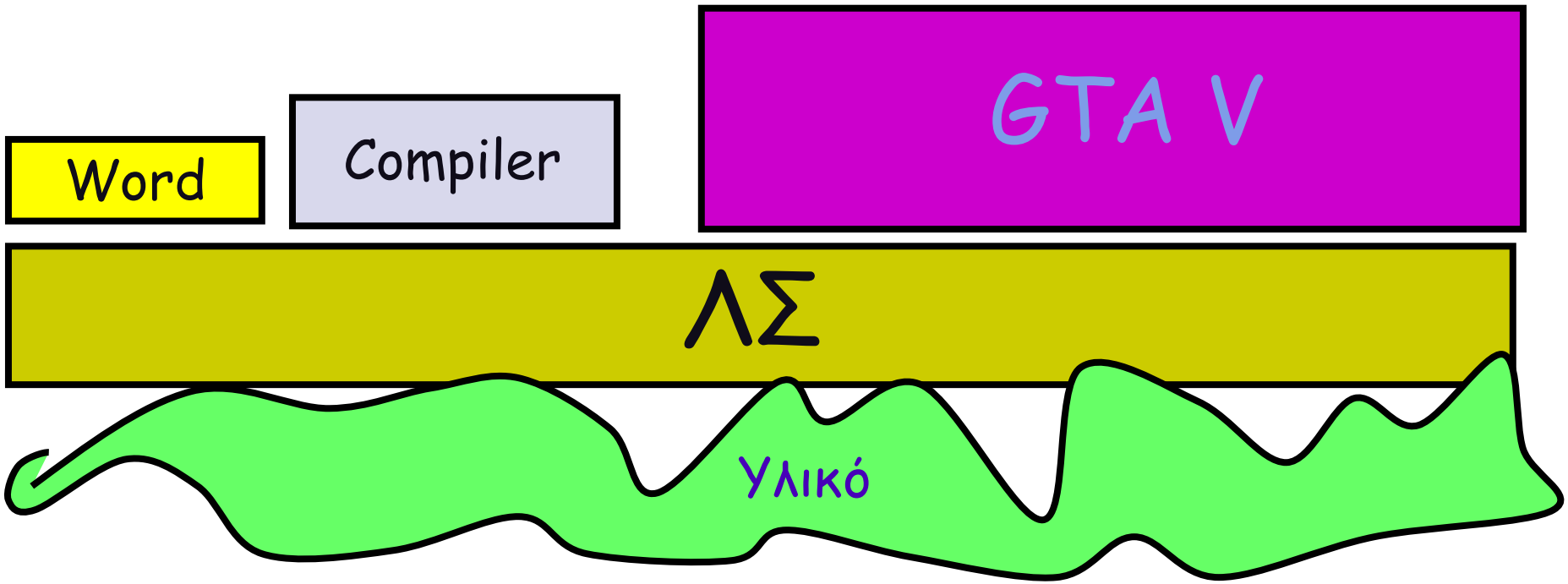
    Γραφείο: Γκλαβάνη, Β4/8

    E-mail: vasiliad@e-ce.uth.gr

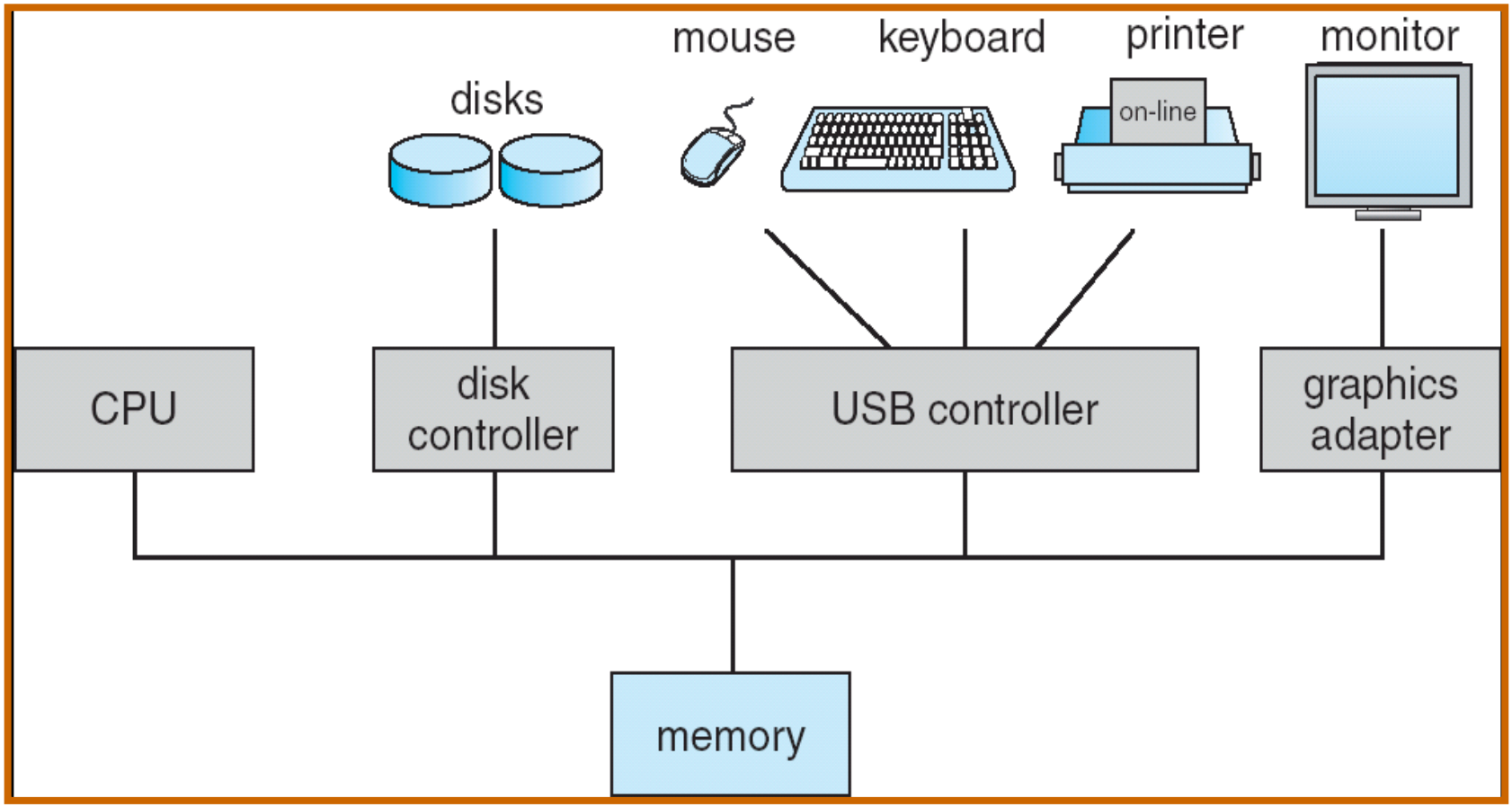
Πρόγραμμα

- Μάθημα (Σαράτση)
  - Δευτέρα 15:00 – 17:00
  - Πέμπτη 15:00 – 17:00
- Φροντιστήριο (Σ)
  - Παρασκευή 9:00 – 10:00

Τα Στοιχεία ενός Υπολογιστικού Συστήματος



Υλικό ενός Τυπικού ΥΣ

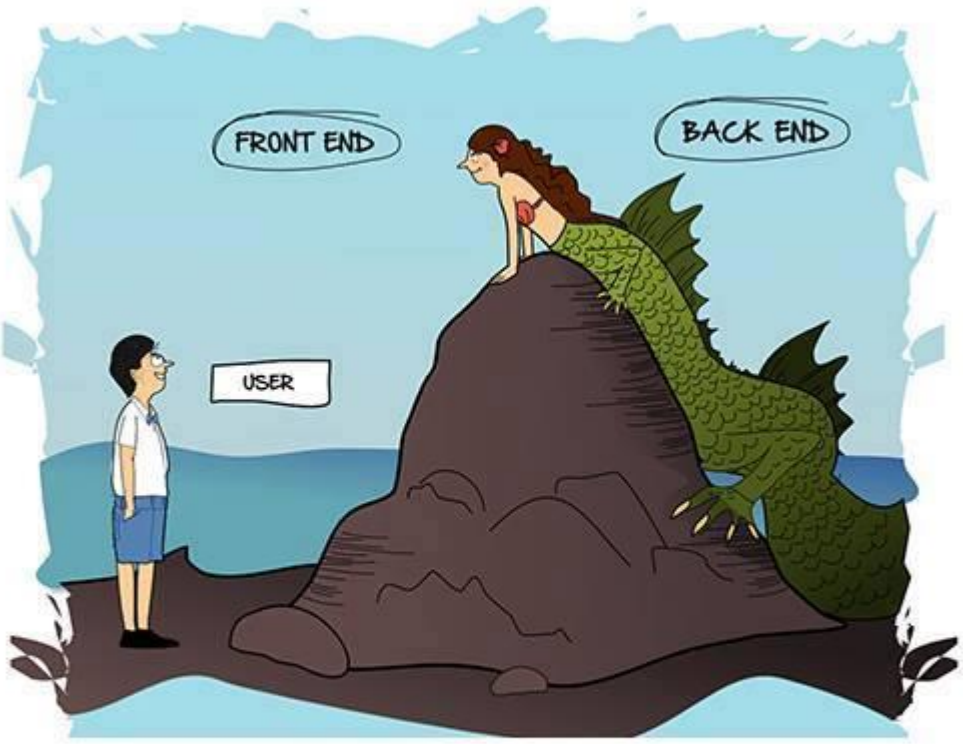


Τι είναι το Λειτουργικό Σύστημα (ΛΣ);

- Ένα πρόγραμμα που λειτουργεί ως *ενδιάμεσος* μεταξύ των χρηστών των Υπολογιστικών Συστημάτων και του υλικού του Υπολογιστικού Συστήματος (ΥΣ)

Τι το ενδιαφέρον υπάρχει εδώ;

- ΛΣ = Η «λάσπη» που δένει το υλικό με το λογισμικό και η «κουρτίνα» που κρύβει το υλικό από το λογισμικό.
  - Κάνει «ευπαρουσίαστη» την πραγματικότητα
  - Μαγεία για τον πολύ κόσμο…



Front end vs. Back end.

Τι το ενδιαφέρον υπάρχει εδώ;

- ΛΣ = Παράδειγμα ενός πολύ πολύπλοκου συστήματος λογισμικού
  - Μεγάλο, Παράλληλο, Ακατανόητο, Πανάκριβο (για να δημιουργηθεί)
    - Windows (τουλάχιστον ως τα XP): 10 χρόνια 1000δες προγραμματιστών. Δουλεύει;
  - Τα ενδιαφέροντα συστήματα είναι και πολύπλοκα: internet, έλεγχος εναέριας κυκλοφορίας, κυβερνήσεις, καιρός, ζευγάρια ☺, …
- Πώς αντιμετωπίζουμε την πολυπλοκότητα;
  - Αφαίρεση + Δομή + Επαναχρησιμοποίηση
  - Εξαιρετικά αποτελεσματικό:

```
int main(int argc, char *argv[]) {
    printf("Hello World");
    return(0);
}
```

  - Χιλιάδες γραμμές κώδικα! Αλλά δε χρειάζεται να νοιαστούμε…

Απόπειρες Ορισμού ΛΣ (1/3)

- **Το ΛΣ ως εκτεταμένη μηχανή (extended ή virtual machine).**
  - Το πρόγραμμα που κρύβει από το χρήστη / προγραμματιστή την αλήθεια για το υλικό
  - **Παράδειγμα 1:** Παρουσίαση μιας απλής και εύχρηστης απεικόνισης από ονόματα και λειτουργίες χειρισμού αρχείων και καταλόγων
  - **Παράδειγμα 2:** Παρουσίαση της μνήμης στα προγράμματα των χρηστών, διαχείριση διακοπών (interrupt handling)

Απόπειρες Ορισμού ΛΣ (2/3)

- **Το ΛΣ ως διαχειριστής της ανάθεσης πόρων (resource allocation).**
  - Το πρόγραμμα που αναλαμβάνει να μοιράσει τους πόρους τους συστήματος ανάμεσα στις διάφορες εφαρμογές
  - Το πρόγραμμα που κάνει το πεπερασμένο (σχεδόν) άπειρο
  - **Παράδειγμα 1:** Η χρήση κοινών εκτυπωτών, όπου θα πρέπει το ΛΣ να παρέχει έναν τρόπο για την ορθή και με συγκεκριμένη σειρά εκτύπωση των δεδομένων όλων των χρηστών, που χρησιμοποιούν ταυτόχρονα τον εκτυπωτή
  - **Παράδειγμα 2:** Διαχείριση και προστασία της μνήμης, ιδιαίτερα σε συστήματα που εξυπηρετούν ταυτόχρονα πολλούς χρήστες

Απόπειρες Ορισμού ΛΣ (3/3)

- **Top down view:** Παρέχει στα προγράμματα εύκολη και αποδοτική επικοινωνία με τους διάφορους πόρους του ΥΣ
- **Bottom up view:** «Παρέχει μια συστηματοποιημένη και ελεγχόμενη κατανομή των επεξεργαστών, των μνημών, και των άλλων συσκευών εισόδου / εξόδου, ανάμεσα στα διάφορα προγράμματα-πελάτες που ανταγωνίζονται μεταξύ τους για να τα χρησιμοποιήσουν» (Tanenbaum, 2001)

Υλοποίηση Λειτουργικών Συστημάτων

- Στα αρχαία χρόνια σε assembly. Πλέον σε γλώσσες προγραμματισμού υψηλότερου επιπέδου
- Γιατί;
  - μπορεί να γραφτεί γρηγορότερα
  - είναι περισσότερο συμπαγή
  - είναι εύκολα στην κατανόηση
  - είναι πιο εύκολα στην εκσφαλμάτωση (debugging)
  - είναι ευκολότερα μεταφέρσιμα σε άλλη αρχιτεκτονική

Λειτουργικά Συστήματα και Αρχιτεκτονική Υπολογιστικών Συστημάτων

- Σχέση «εξάρτησης» μεταξύ ΛΣ και αρχιτεκτονικής των ΥΣ στα οποία εκτελούνται
- Οι εξελίξεις στο υλικό των ΥΣ=> Επιπλέον λειτουργίες προς τα προγράμματα των χρηστών.
  - Οι λειτουργίες αυτές υλοποιούνταν με την εισαγωγή και την εξέλιξη των ΛΣ
- Καλός τρόπος να κατανοήσουμε τις εξελίξεις στα ΛΣ είναι να παρακολουθήσουμε τις εξελίξεις στην αρχιτεκτονική που τις «επέτρεψαν»

Γενιά 0

- Ένα πρόγραμμα, ένας χρήστης, μια μηχανή
  - Παραδείγματα:
    - Πρώτοι υπολογιστές
    - Πρώτα PCs
    - Nintendo, Xbox, PS3
    - Αυτοκίνητο
    - Ανελκυστήρας

- Απλά μια βιβλιοθήκη standard υπηρεσιών (device drivers, interrupt handlers, I/O)
- Τα καλά:
  - Δεν υπάρχουν κακοί (άνθρωποι & προγράμματα)
  - Ελάχιστες πολύπλοκες αλληλεπιδράσεις
- Τα άσχημα:
  - Κακή αξιοποίηση των πόρων

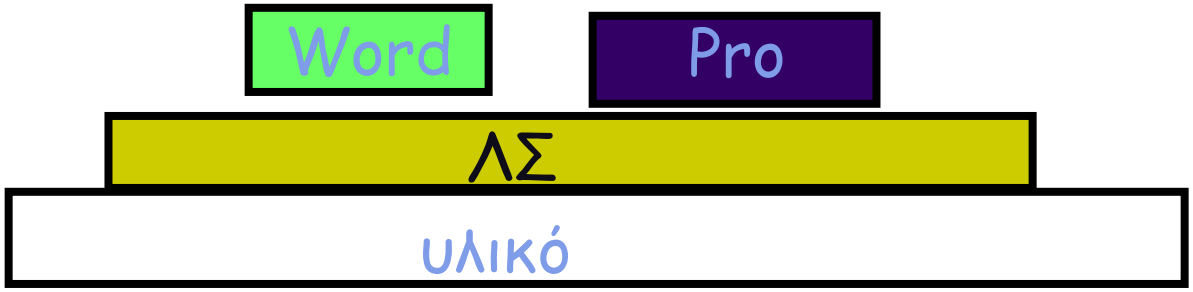
Μεγάλα Υπολογιστικά Συστήματα

- Μείωση του χρόνου εκκίνησης με την ομαδοποίηση παρόμοιων εργασιών (batching of similar jobs)
- Αυτοματοποίηση της εναλλαγής μεταξύ των εργασιών – αυτόματη μεταφορά ελέγχου μεταξύ εργασιών (πρώτο στοιχειώδες ΛΣ)
- Μόνιμο (στη μνήμη) πρόγραμμα ελέγχου (resident monitor):
  - Αρχικά, ο έλεγχος στο πρόγραμμα ελέγχου
  - Μεταφορά του ελέγχου στην εργασία
  - Με την ολοκλήρωση της εργασίας, μεταφορά του ελέγχου στο πρόγραμμα ελέγχου

Σειριακή Εκτέλεση και Παροχέτευση (Spooling)

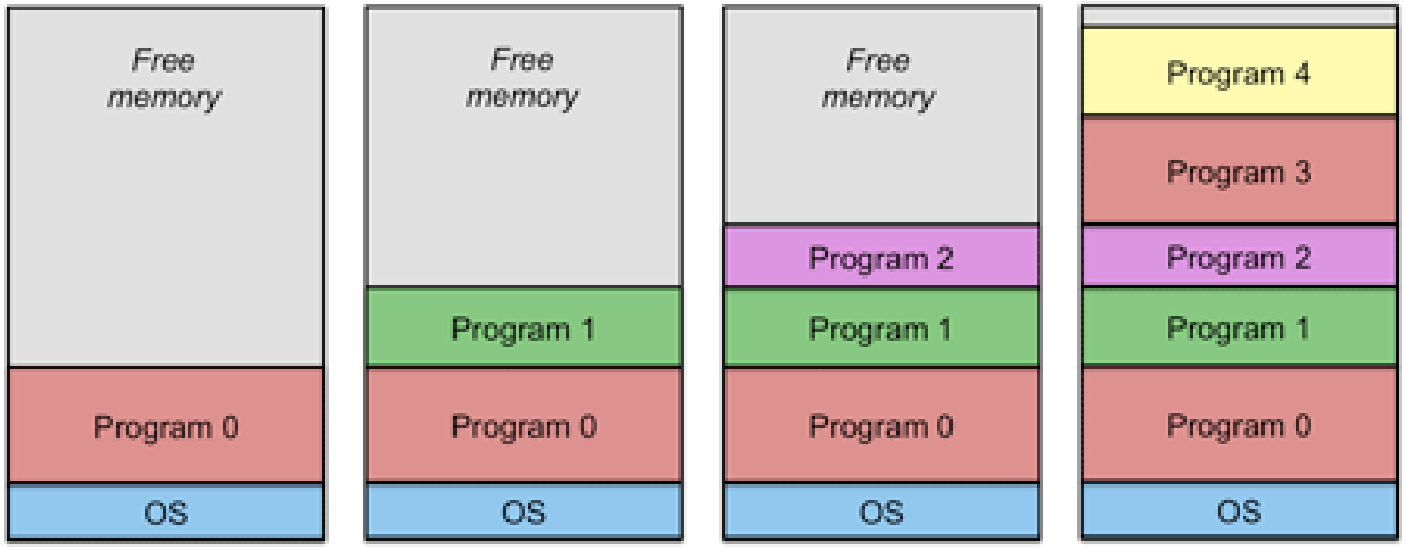
- Οι εργασίες προς εκτέλεση φορτώνονται σε δευτερεύουσα μνήμη
  - Οι αρχαίοι ημών πρωτόγονοι έγραφαν κάθε εργασία σε χάρτινες κάρτες, οι οποίες με την σειρά τους διαβάζονταν από ειδικά μηχανήματα για να αποθηκευτούν στον δίσκο/ταινία του ΥΣ
- Το σύστημα επιλέγει μια από τις αποθηκευμένες εργασίες, την μεταφέρει από την δευτερεύουσα στην πρωτεύουσα μνήμη και αρχίζει την εκτέλεσή της
- Μόλις τελειώνει η εκτέλεση της τρέχουσας εργασίας, το ΛΣ φορτώνει την επόμενη εργασία
- Κάθε χρονική στιγμή, υπάρχει μόνο μια εργασία που είναι φορτωμένη και εκτελείται από το σύστημα

Γενιά 1



- Το απλό ΛΣ δεν είναι αποδοτικό:
  - Αν μια διεργασία περιμένει κάτι, η μηχανή «χαζεύει».
- (Φαινομενικά) Απλή ιδέα:
  - Τρέξε ταυτόχρονα περισσότερες διεργασίες
  - Όταν κάποια περιμένει, τρέξε άλλη
- 1-2 προβληματάκια: Τι κάνουμε αν ένα πρόγραμμα
  - Πέσει σε άπειρο loop;
  - Αρχίσει να ακουμπάει ολοένα και περισσότερη μνήμη;

Χάρτης Μνήμης σε Πολυπρογραμματισμένο Σύστημα



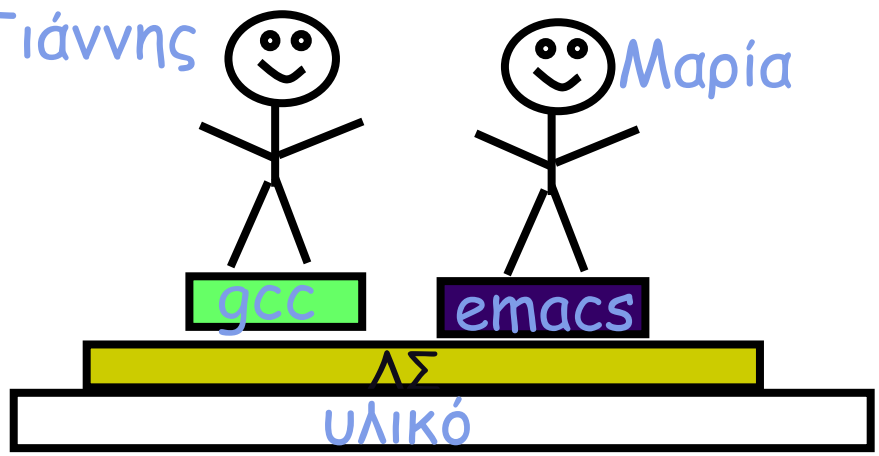
Χαρακτηριστικά ΛΣ για υποστήριξη πολυπρογραμματισμού (1/2)

- Διαχείριση μνήμης:
  - Το ΛΣ πρέπει να είναι σε θέση να χωρίσει τη μνήμη σε τμήματα (ένα για κάθε διεργασία) και να προστατεύσει το τμήμα κάθε διεργασίας από (ηθελημένες ή αθέλητες) παρεμβολές των υπολοίπων διεργασιών
- Διαχείριση διεργασιών:
  - Το ΛΣ πρέπει να είναι σε θέση να επιλέξει ποιες διεργασίες θα αποκτήσουν χώρο στη μνήμη
    - Θεωρούμε ότι η μνήμη δεν επαρκεί για να «στεγάσει» όλες τις διεργασίες ταυτόχρονα και ότι μόνο μια διεργασία που είναι στη μνήμη μπορεί να τρέξει

Χαρακτηριστικά ΛΣ για υποστήριξη πολυπρογραμματισμού (2/2)

- Χρονοπρογραμματισμός της CPU:
  - Το ΛΣ πρέπει να επιλέξει μεταξύ των διεργασιών που έχουν χώρο στη μνήμη κάποια για να τρέξει
- Ανάθεση πόρων με τέτοιο τρόπο ώστε να μην «επηρεάζεται» η μια διεργασία από την εκτέλεση άλλων διεργασιών

Γενιά 2



- Το απλό ΛΣ καταντά ακριβό:
  - 1 χρήστης = 1 υπολογιστής
- (Φαινομενικά) Απλή Ιδέα:
  - Επέτρεψε παραπάνω από 1 χρήστη στο σύστημα.
  - Τρέχει το σύστημα Ν φορές πιο αργά; Συνήθως όχι!
  - Παρατήρηση: Οι χρήστες λειτουργούν με εξάρσεις. Αν κάποιος είναι ανενεργός, δώσε τους πόρους στον άλλο.
- 1-2 προβληματάκια:
  - Τι κάνουμε αν οι χρήστες είναι «σπάταλοι»;
  - Κακοί;
  - Η απλά πολλοί;
- Το ΛΣ προσθέτει (ξανά) προστασία
  - (Προσοχή: Όσο πάμε να βελτιώσουμε την αξιοποίηση πόρων, αυξάνουμε την πολυπλοκότητα)

Όταν κοιτάζω από ψηλά… (προστασία)

- Στόχος: Απομόνωση «κακών» προγραμμάτων και χρηστών (ασφάλεια)
  - Εργαλεία: Διακοπή (preemption) + Μεσολάβηση (interposition) + Προνομιούχες (privileged) εντολές
- Διακοπή:
  - Δίνουμε κάτι στην εφαρμογή – μπορούμε όμως να της το πάρουμε πίσω
- Μεσολάβηση:
  - ΛΣ ανάμεσα στην εφαρμογή και στα “καλούδια” (πόρους)
  - Καταγραφή όλων των πόρων που η εφαρμογή μπορεί να προσπελάσει
  - Σε κάθε προσπέλαση έλεγχος αν είναι νόμιμη
- Προνομιούχα / μη προνομιούχα εκτέλεση
  - Εφαρμογές μη προνομιούχες (πρόβατα)
  - ΛΣ προνομιούχο (Θεός)
  - Οι διαδικασίες προστασίας ελέγχονται μόνο κατά την προνομιούχα εκτέλεση

Παραδείγματα Επιτυχημένων Μεθόδων Προστασίας

- Προστασία του επεξεργαστή: Διακοπή (preemption)
  - Διακοπή (interrupt) ρολογιού: Το υλικό περιοδικά διακόπτει την εφαρμογή και μεταφέρει τον έλεγχο στο ΛΣ
  - Το ΛΣ αποφασίζει αν θα πάρει τον επεξεργαστή από την εφαρμογή
  - Άλλες περιπτώσεις; Η διεργασία μπλοκάρει, ολοκληρώνεται μια ενέργεια I/O, κλήση συστήματος

- Προστασία μνήμης : Μετάφραση διευθύνσεων
  - Ελέγχεται η νομιμότητα όλων των αναγνώσεων και εγγραφών.
  - Χρήση μηχανισμού υλικού για τη μετάφραση διευθύνσεων (γιατί;)
  - (Παρόμοια και η προστασία του δίσκου)

Τρύπες σε Πραγματικά Συστήματα

- Τα ΛΣ προστατεύουν κάποιες ενέργειες αλλά αγνοούν (ηθελημένα ή μη) άλλες.
- Τα περισσότερα UNIX θα «σκάσουν» αν τρέξετε αυτό

```
int main() {
    while(1)
        fork();
}
```

Συνήθης απόκριση: Πάγωμα

Για ξεπάγωμα: Reboot

- Αν αυτό επιτύχει, δοκιμάστε να «ακουμπάτε» ολοένα και περισσότερη μνήμη
- Τεχνικές ή «κοινωνικές» λύσεις:
  - Τεχνική: Όρια (quotas)
  - Κοινωνική: Τιμωρία του χαζού χρήστη

Do not try this at home.
Do not try this at the Dept. !!!

Το θαύμα της Κανά (1/2)

- Περιορισμένη «πίτα» (πόροι) – Άπειρη ζήτηση
- Πώς κάνουμε την πίτα να φτάσει;
  - Ιδέα: Η χρήση πόρων έχει εξάρσεις! Δώσε στους άλλους όταν κάποιος είναι ανενεργός
    - Π.χ. περιμένουμε μια web page; Δώσε τη CPU σε άλλη διεργασία
    - Νέα ιδέα; Μοιραζόμαστε αίθουσες, εστιατόρια, μέσα μεταφοράς, δρόμους αντί να έχει ο καθένας το δικό του
- ΟΜΩΣ, Μεγαλύτερη αξιοποίηση = Παραπάνω πολυπλοκότητα.
  - Διαχείριση; (π.χ., 1 δρόμος / αυτοκίνητο αντί της εθνικής οδού)
  - Αφαίρεση (λωρίδες), συγχρονισμός (φανάρια), αύξηση χωρητικότητας (περισσότεροι δρόμοι)
- ΟΜΩΣ, Μεγαλύτερη αξιοποίηση = Ισχυρότερος ανταγωνισμός. Τι κάνουμε όταν η ψευδαίσθηση χαλάσει;
  - Άρνηση υπηρεσίας (καλέστε αργότερα), εφεδρείες (VM swapping), υποχώρηση και επανάληψη (ethernet), Houston we have a problem (εθνική οδός)

Το θαύμα της Κανά (2/2)

- Πώς μοιράζουμε την «πίτα»;
  - Να το κάνει ο χρήστης;
  - Θεωρούμε τις εφαρμογές ισότιμες, τις παρακολουθούμε και προσαρμοζόμαστε
- Τι να θυμάστε;
  - ΛΣ = Ο δικτάτορας του υπολογιστή
  - Χρησιμοποίησε πληροφορία από το σύστημα αντί να είσαι τυφλά δίκαιος
- Και τι κάνουμε με τους κακούς;
  - Όρια (quotas), στην άκρη (swapping), αγοράζουμε περισσότερα (microsoft windows), τίποτα-καταστροφή (ethernet, τα περισσότερα πραγματικά συστήματα), νόμοι (εθνική οδός)
  - Πρόβλημα: Δύσκολο να ξεχωρίσεις τα καλά, πλην όμως «βαριά» προγράμματα, από τους εγωιστές ή τους κακούς.

Τρέχουμε … (;) (επίδοση)

- Κόλπο 1: Αξιοποίησε τις εξάρσεις
  - Πάρε από τον ανενεργό και δώσε στον ενεργό. Τους έχεις όλους ευτυχισμένους
- Κόλπο 2: Αξιοποίησε την «τάση»
  - 80% του χρόνου από 20% του κώδικα
  - 90% των αναφορών σε 10% της μνήμης
  - Η ιδέα πίσω από την cache: Βάλε 10% στη γρήγορη μνήμη, 90% στην αργή, και φαίνεται σαν μια μεγάλη γρήγορη μνήμη
- Κόλπο 3: Το παρελθόν προβλέπει το μέλλον
  - Ποια γραμμή να αντικαταστήσω στην cache; Αν παρελθόν=μέλλον, αυτή που χρησιμοποιήθηκε παλαιότερα.
  - ΔΟΥΛΕΥΕΙ: καιρός, χρηματιστήριο, …

Στόχοι Σχεδιασμού ΛΣ

- Στόχοι για το Χρήστη
  - Το ΛΣ πρέπει να είναι εύκολο στη χρήση, εύκολο στη μάθηση, αξιόπιστο, ασφαλές και γρήγορο
- Στόχοι για το Σύστημα
  - Εύκολο στο σχεδιασμό, την υλοποίηση και τη συντήρηση,
  - Ευέλικτο, αξιόπιστο, χωρίς λάθη και αποδοτικό

Μηχανισμοί και Πολιτικές

- **Μηχανισμοί**
  - Προσδιορίζουν το **πως** υλοποιούνται συγκεκριμένες λειτουργίες του ΛΣ
- **Πολιτικές**
  - Προσδιορίζουν το **τι** τακτική θα ακολουθηθεί σε μια συγκεκριμένη λειτουργία του ΛΣ
  - Υλοποιούνται από μηχανισμούς
- Μέσα από έναν μοναδικό μηχανισμό μπορεί να υποστηριχθούν πολλές διαφορετικές πολιτικές
- Ο διαχωρισμός της πολιτικής από τους μηχανισμούς είναι σημαντική αρχή (ευελιξία)

Δομή Λειτουργικών Συστημάτων

Χρήστος Δ. Αντωνόπουλος
2/3/2017

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μονολιθική Προσέγγιση

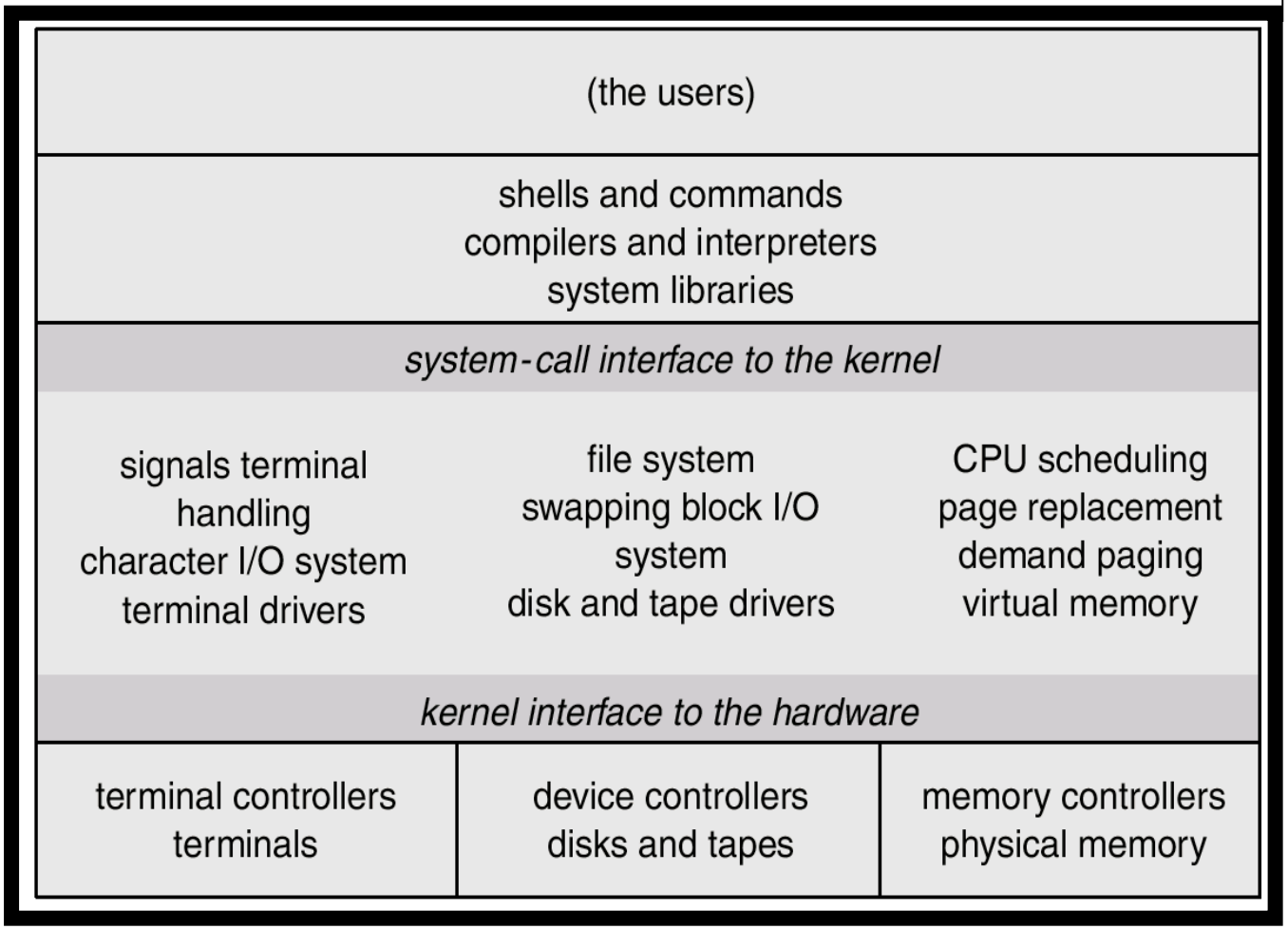
- Το ΛΣ αποτελείται από ουσιαστικά ένα μεγάλο κομμάτι κώδικα
- Πρόσβαση στις λειτουργίες του ΛΣ δίνεται μέσω από μια μοναδική και αρκετά εκτεταμένη διεπαφή προγραμματισμού (Application Programming Interface - API)
- Ακόμα και αν υπάρχουν ξεχωριστά τμήματα κώδικα (που μεταφράζονται ξεχωριστά), υπάρχει αρκετά μεγάλη εξάρτηση ανάμεσα τους (σε επίπεδο δομών δεδομένων ή/και αλγορίθμων)
- Για να αλλάξει μια λειτουργία του ΛΣ πρέπει συνήθως να γίνουν αλλαγές σε πολλά διαφορετικά τμήματα του κώδικα

Δομή του UNIX

UNIX – περιορισμένο από τις δυνατότητες του υλικού, αρχικά το ΛΣ UNIX είχε περιορισμένη δόμηση. Το UNIX αποτελείται από δύο διαχωρίσιμα τμήματα:

- Τον πυρήνα (kernel):
  - Αποτελείται από οτιδήποτε υπάρχει κάτω από τη διεπαφή των κλήσεων συστήματος και πάνω από το υλικό
  - Παρέχει τις λειτουργίες συστήματος αρχείων, διαχείρισης μνήμης, χρονοπρογραμματισμού ΚΜΕ, και άλλες λειτουργίες ενός ΛΣ (μεγάλος αριθμός λειτουργιών για ένα επίπεδο)
- Προγράμματα συστήματος

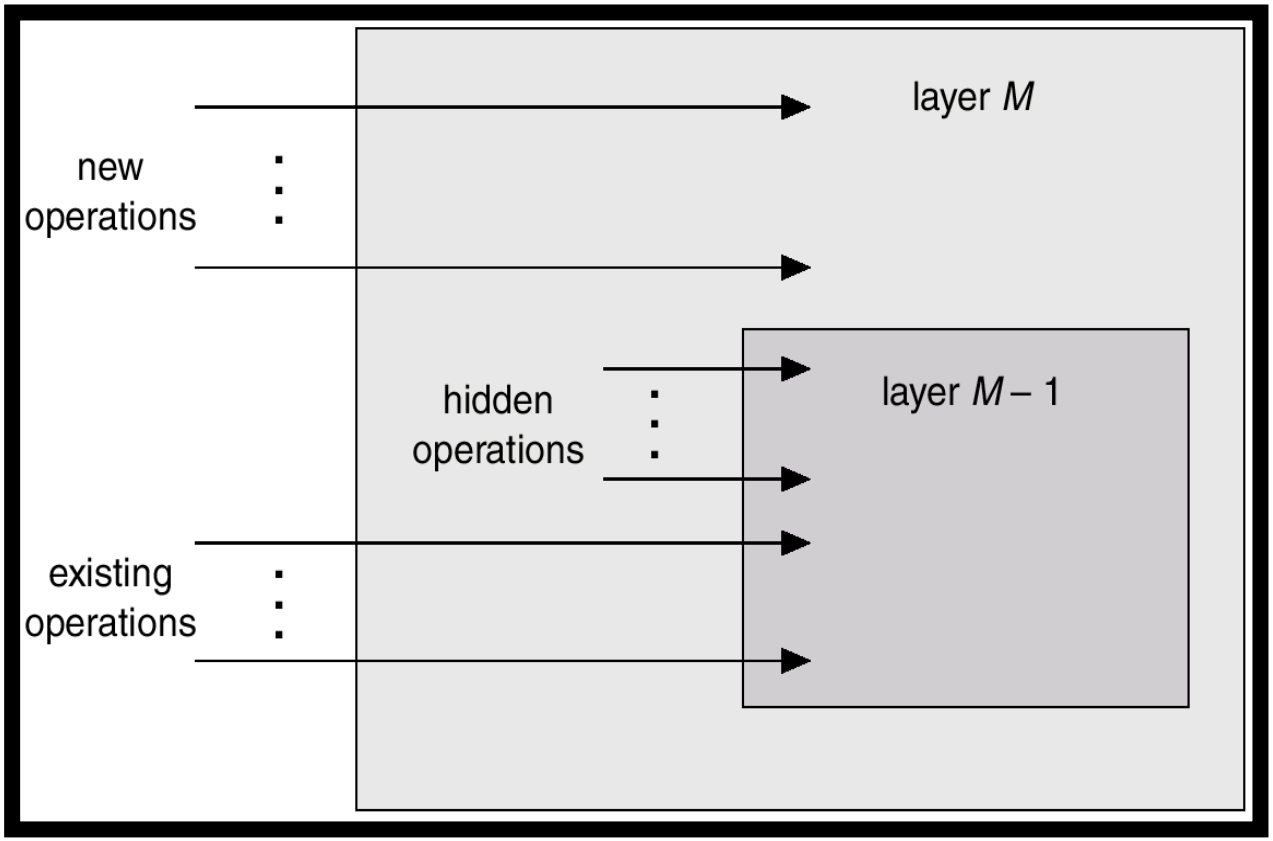
Δομή του Unix



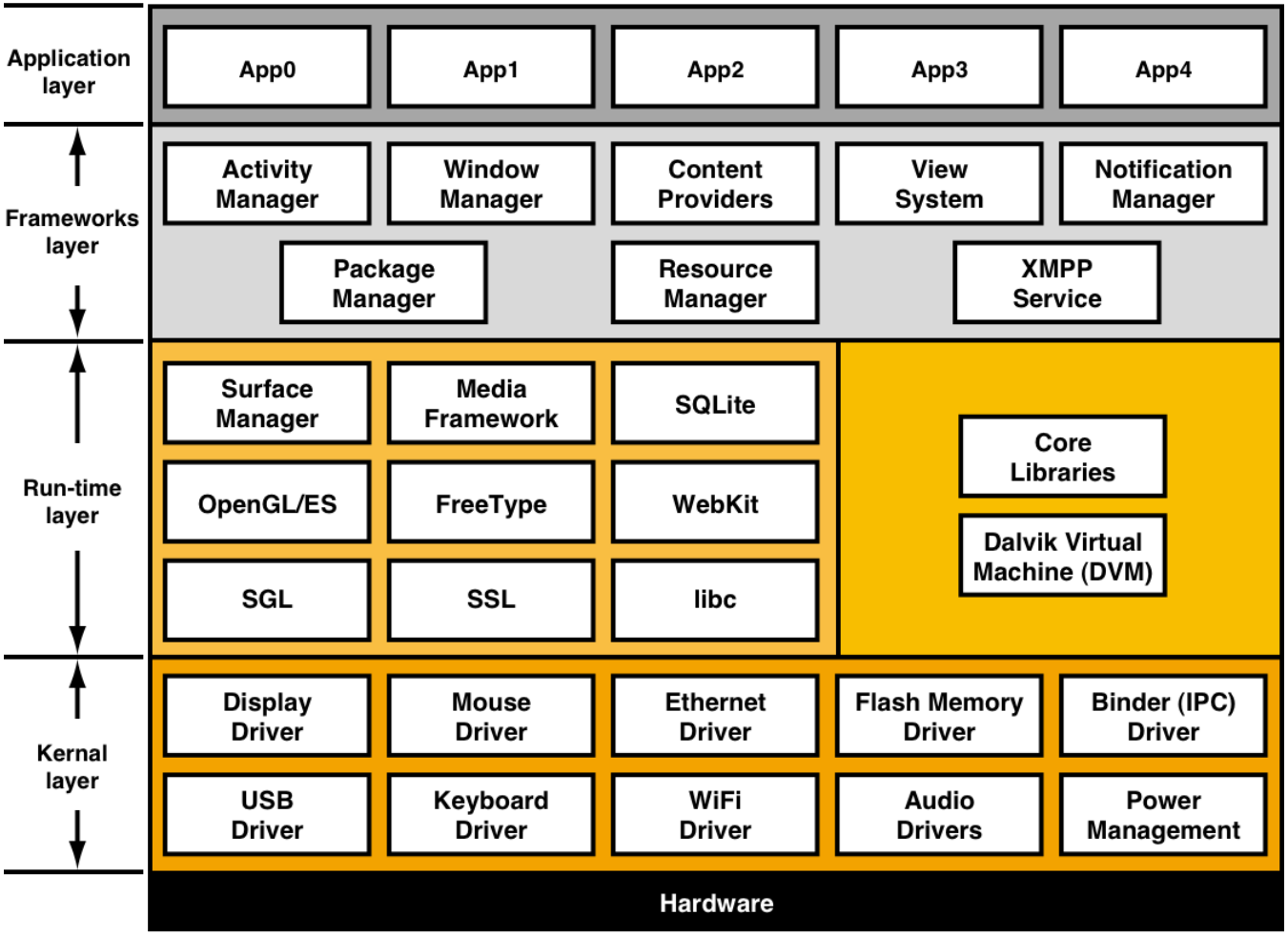
Προσέγγιση σε Επίπεδα (Layered Approach)

- Το ΛΣ χωρίζεται σε επίπεδα-στρώματα (layers ή levels),
- Κάθε ένα χτίζεται πάνω από υπάρχοντα επίπεδα. Επίπεδο 0 = υλικό, ενώ το υψηλότερο (επίπεδο N) η διεπαφή επικοινωνίας χρήστη-υπολογιστή (user interface)
- Με την τμηματοποίηση (modularity), τα επίπεδα επιλέγονται έτσι ώστε το καθένα να χρησιμοποιεί λειτουργίες και υπηρεσίες μόνο από τα χαμηλότερα επίπεδα

Αναπαράσταση Στρώματος



Android OS (layered)

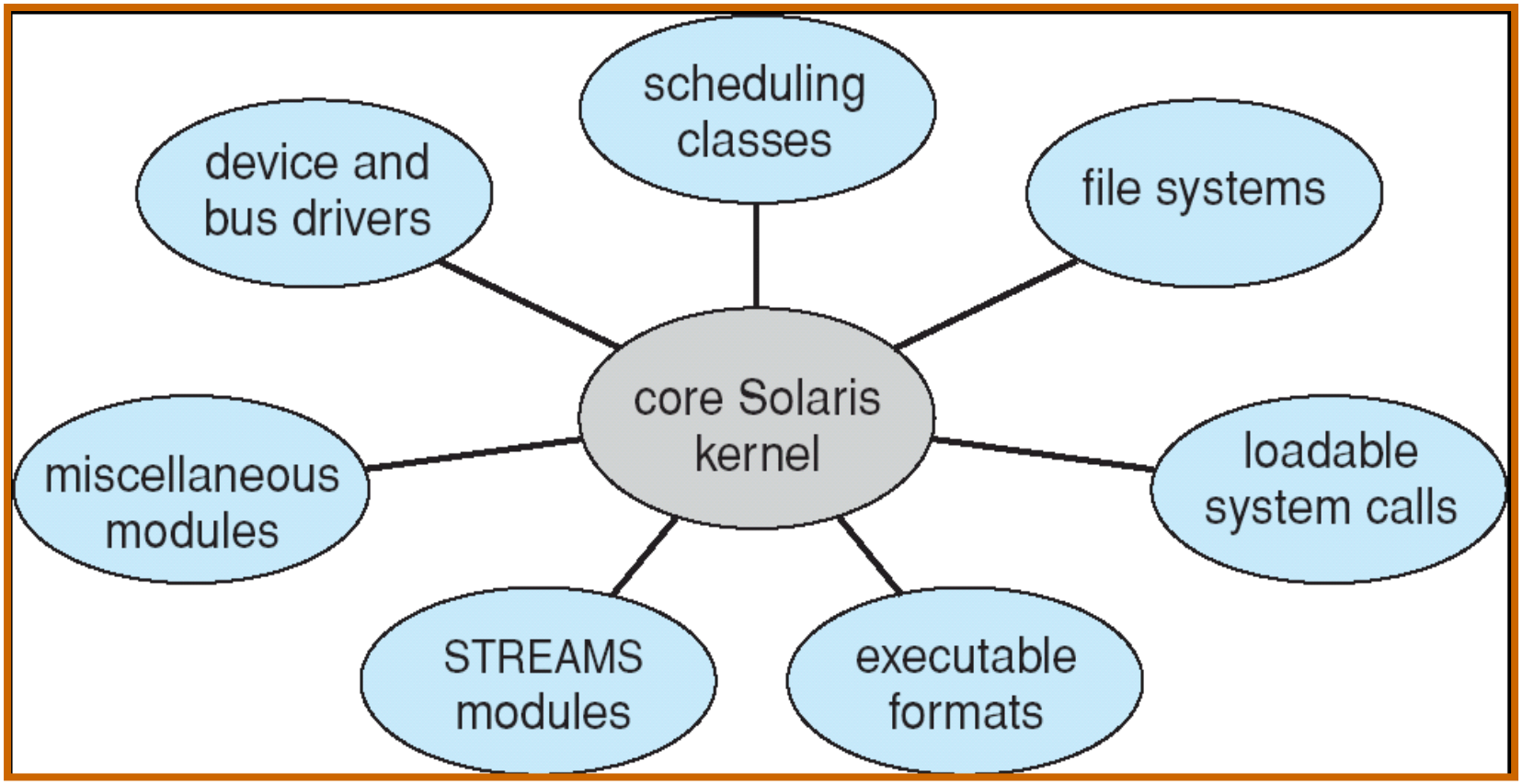


C, C++, native code	Java
= Linux Kernel	= Android frameworks
= Libraries	= Applications
= Android runtime	

Δομικά Τμήματα (Modules)

- Τα περισσότερα σύγχρονα λειτουργικά συστήματα υλοποιούν δομικά τμήματα πυρήνα (kernel modules)
  - Χρησιμοποιείται η αντικειμενοστραφής προσέγγιση
  - Το κάθε τμήμα είναι ξεχωριστό
  - Το κάθε ένα μιλάει στο άλλο μέσα από γνωστές διεπαφές (interfaces)
  - Το κάθε τμήμα μπορεί να φορτωθεί στον πυρήνα με βάση τις ανάγκες
- Συνολικά, παρόμοιο με τα επίπεδα αλλά περισσότερο ευέλικτο

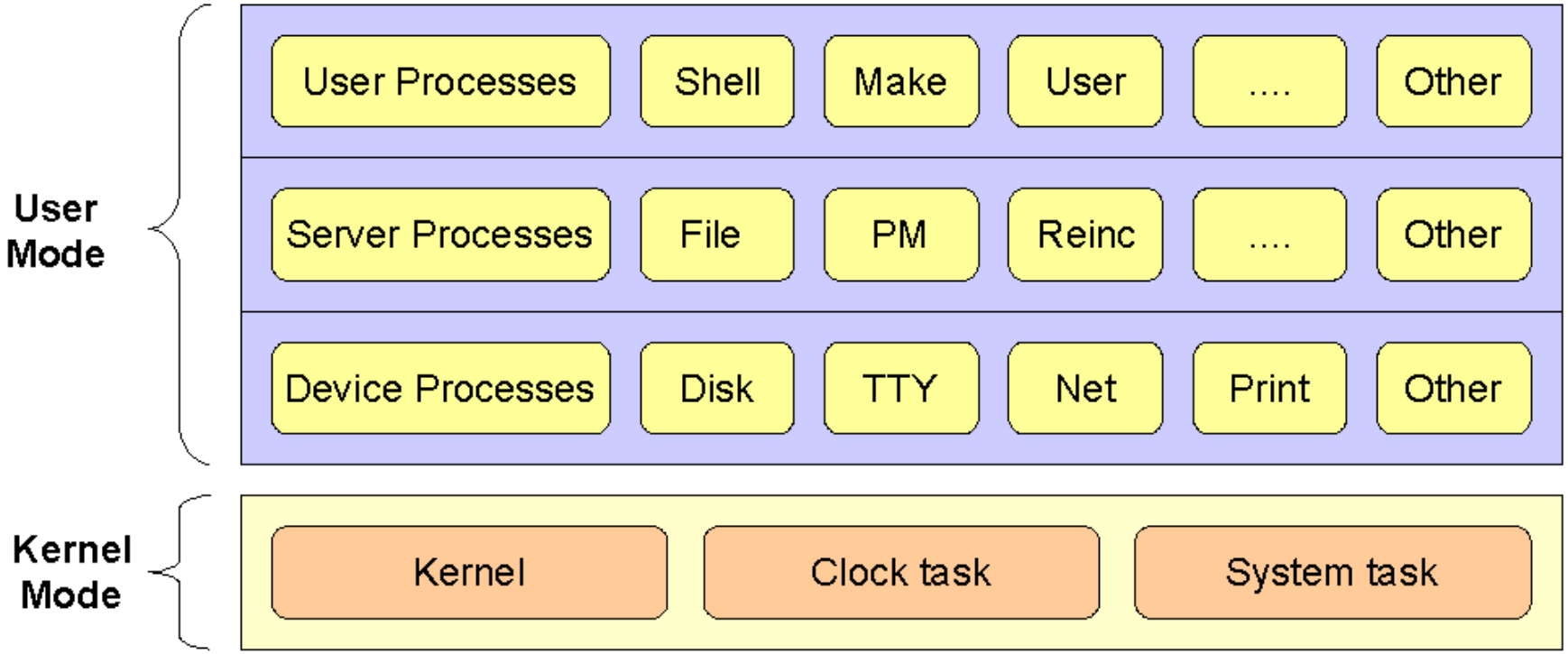
Δομικά Τμήματα (συνέχεια)



Προσέγγιση Microkernel

- Μετακινεί όσο το δυνατόν περισσότερες λειτουργίες στο χώρο του χρήστη
- Η επικοινωνία μεταξύ των τμημάτων των χρηστών γίνεται με πέρασμα μηνυμάτων
- Πλεονεκτήματα:
  - ευκολότερη και πιο ευέλικτη επεκτασιμότητα του ΛΣ
  - ευκολότερη η μεταφορά του ΛΣ σε νέες αρχιτεκτονικές
  - μεγαλύτερη αξιοπιστία και ασφάλεια (λιγότερος κώδικας τρέχει σε kernel mode)

Minix Microkernel Architecture

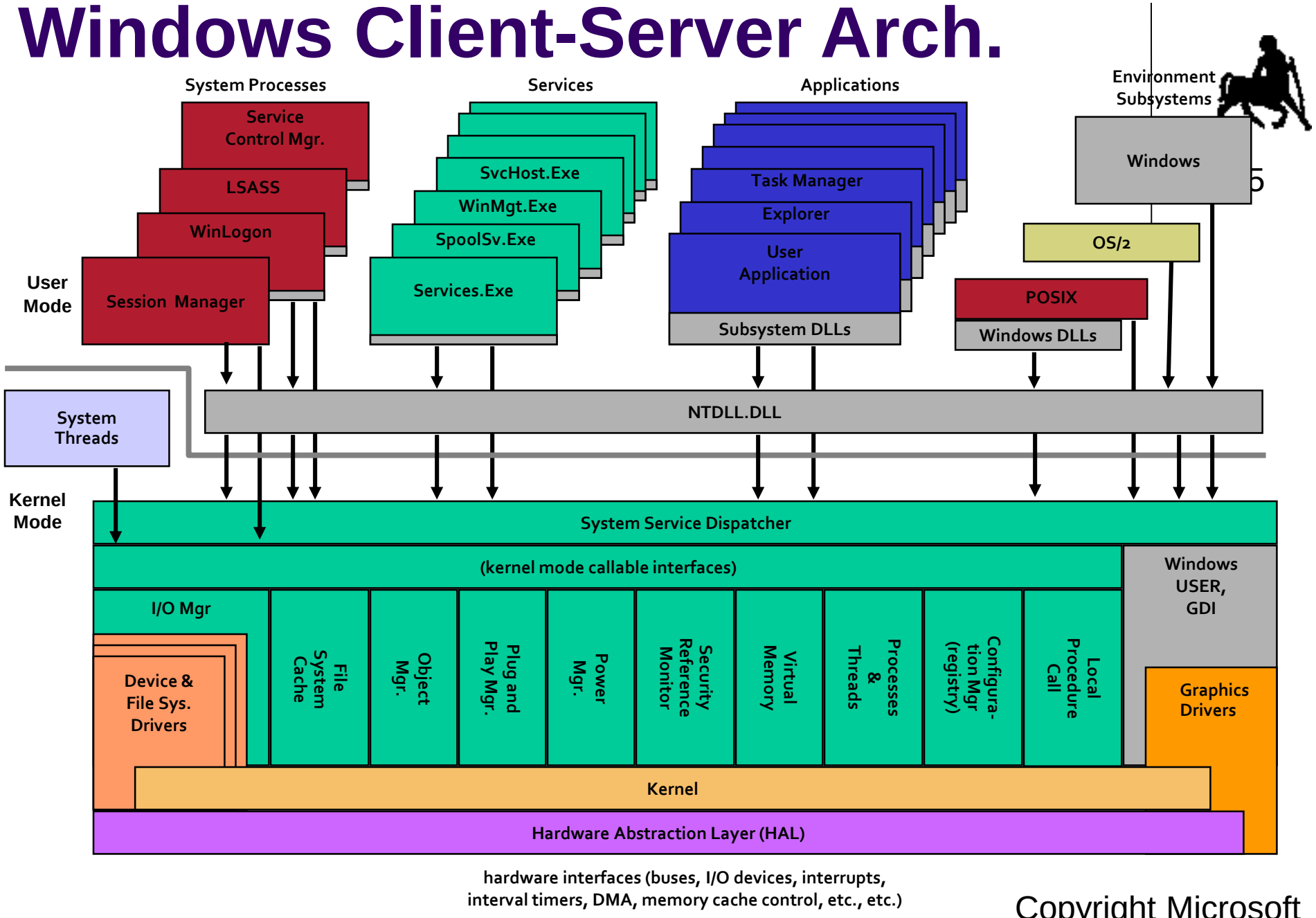


The MINIX 3 Microkernel Architecture

Προσέγγιση Client-Server

- Το ΛΣ αποτελείται από μια ομάδα εξυπηρετητών
- Κάθε εξυπηρετητής του ΛΣ ανοίγει ένα κανάλι / δίαυλο επικοινωνίας και περιμένει να λάβει αιτήσεις από πελάτες (προγράμματα)
- Ένας εξυπηρετητής είναι είτε σειριακός (iterative) όπου επεξεργάζεται μια αίτηση κάθε φορά (οι αιτήσεις των πελατών τοποθετούνται σε ουρές αναμονής), είτε παράλληλος (concurrent) όπου επεξεργάζεται πολλές αιτήσεις ταυτόχρονα
- Τα προγράμματα πελάτες επικοινωνούν με το ΛΣ στέλνοντας αιτήσεις στις διευθύνσεις των αντίστοιχων εξυπηρετητών και λαμβάνοντας τα αποτελέσματα

Windows Client-Server Arch.



hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)

Χρήστος Δ. Αντωνόπουλος
2/3/2017
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

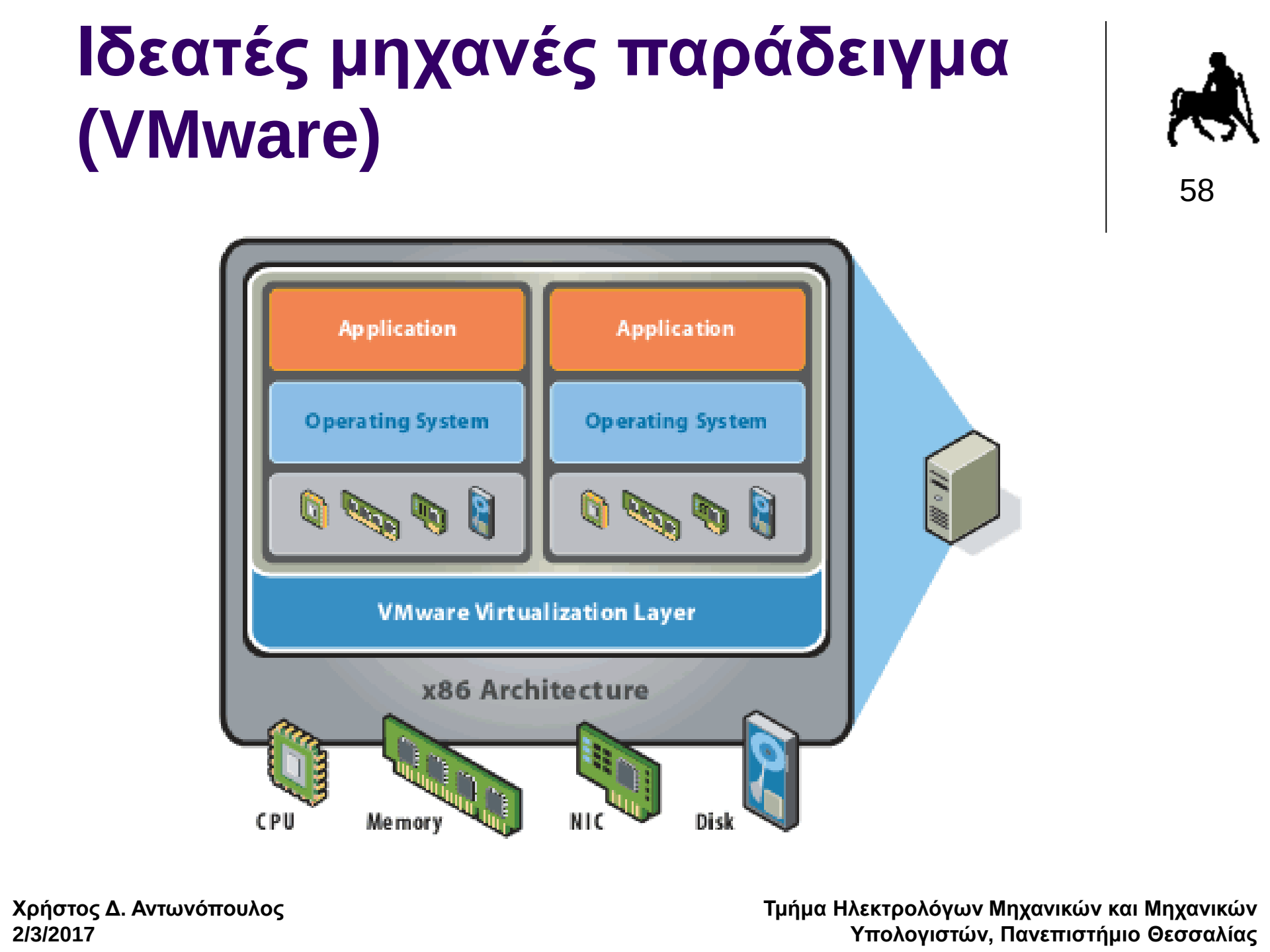
Ιδεατές Μηχανές (Virtual Machines)

- Μια *ιδεατή μηχανή (virtual machine)* παρουσιάζει το υλικό και τον πυρήνα του ΛΣ σαν να ήταν όλα υλικό
- Μια ιδεατή μηχανή παρέχει μια προσαρμογή πανομοιότυπη με το υφιστάμενο υλικό
- Το ΛΣ παρέχει την ψευδαίσθηση των πολλαπλών διεργασιών, όπου η κάθε μια εκτελείται στο δικό της επεξεργαστή με τη δικιά της (ιδεατή) μνήμη

Ιδεατές Μηχανές (συνέχεια)

- Οι πόροι του ΥΣ διαμοιράζονται προκειμένου να δημιουργήσουν τις ιδεατές μηχανές
  - Ο χρονοπρογραμματισμός της ΚΜΕ, μπορεί να δώσει την εντύπωση ότι οι χρήστες έχουν δικό τους επεξεργαστή
  - Ένα κανονικό τερματικό χρήστη σε ένα σύστημα καταμερισμού χρόνου λειτουργεί ως η κονσόλα του χειριστή της ιδεατής μηχανής

Ιδεατές μηχανές παράδειγμα (VMware)



Πλεονεκτήματα / Μειονεκτήματα Ιδεατών Μηχανών

- Πλήρης προστασία των πόρων του συστήματος
  - Κάθε ιδεατή μηχανή είναι πλήρως απομονωμένη από τις άλλες.
  - ΌΜΩΣ: Δεν επιτρέπεται άμεσος διαμοιρασμός πόρων
- Τέλειο κόλπο για έρευνα και ανάπτυξη στην περιοχή των ΛΣ
  - Ανάπτυξη στην ιδεατή μηχανή, αντί της πραγματικής
  - Δεν επηρεάζεται η κανονική λειτουργία του συστήματος
- Δύσκολη στην υλοποίηση λόγω της επιβάρυνσης που απαιτείται