ΜΥΥ601 - Λειτουργικά Συστήματα

Εισαγωγή στο Minix 3.2.0
2η Εργαστηριακή Άσκηση

Γιώργος Καππές

Το Λειτουργικό Σύστημα – Minix

MINIX: MINI uniX

- Αποτελεί ένα εκπαιδευτικό λειτουργικό σύστημα
- Το έγραψε ο Andrew S. Tanenbaum το 1987 βασισμένος στο UNIX
- Σκοπός του είναι η εξοικείωση των φοιτητών με τα λειτουργικά συστήματα
- Είναι εύκολη η κατανόηση του κώδικά του, χωρίς πάντα να πετυχαίνει τη βέλτιστη απόδοση

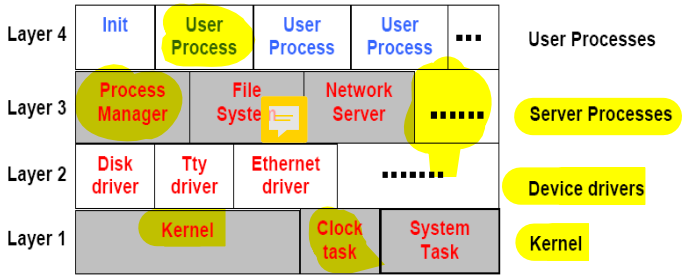
Αρχιτεκτονική

- Το Minix ακολουθεί αρχιτεκτονική **Μικροπυρήνα (Microkernel)**
- Είναι δομημένο με έναν πιο αρθρωτό τρόπο σε σχέση με το Unix
- Ο πυρήνας περιλαμβάνει μόνο πολύ κρίσιμες λειτουργίες. Οι υπόλοιπες υλοποιούνται σε ξεχωριστά τμήματα που εκτελούνται ως διεργασίες χρήστη

Εσωτερική Δομή του Minix

Δομημένο σε 4 επίπεδα

- Επίπεδο 1: Πυρήνας
- Επίπεδο 2: Οδηγοί συσκευών
- Επίπεδο 3: Διεργασίες διακομιστή
- Επίπεδο 4: Διεργασίες χρήστη



Επίπεδο 1

Πυρήνας

- Χειρίζεται τις διακοπές και τις παγίδες
- Αποθηκεύει και ανακτά τους καταχωρητές
- Κάνει βασική χρονοδρομολόγηση
- Περιλαμβάνει τη στοιχειώδη εργασία ρολογιού (clock task) και τη στοιχειώδη εργασία συστήματος (system task)
- Προσφέρει στα ανώτερα επίπεδα ένα μοντέλο ανεξάρτητων ακολουθιακών διεργασιών που επικοινωνούν με μηνύματα
- Διασυνδέεται σε ένα μοναδικό πρόγραμμα που αποτελεί τον πυρήνα

Επίπεδα 2,3

Επίπεδο 2: Οδηγοί συσκευών

- Περιέχει τους οδηγούς συσκευών - διεργασίες για την προσπέλαση των συσκευών Ε/Ε

Επίπεδο 3: Διεργασίες διακομιστή

- Περιέχει διεργασίες διακομιστή
  - δεν προσπελαύνουν απευθείας τις θύρες Ε/Ε
  - περιορίζονται στη μνήμη που τους έχει ανατεθεί
- Διαχειριστής διεργασιών → υλοποιεί κλήσεις συστήματος που σχετίζονται με τη διαχείριση διεργασιών (π.χ. fork)
- Σύστημα αρχείων → υλοποιεί κλήσεις συστήματος αρχείων (π.χ. read, write)
- Διαχειριστής χρονοδρομολόγησης → υλοποιεί διάφορους αλγόριθμους χρονοδρομολόγησης διεργασιών χρήστη

Επίπεδο 4

Διεργασίες χρήστη

- Περιέχει τις διεργασίες χρήστη (φλοιός, κειμενογράφοι κλπ.)
- Περιέχει την init
  - πρώτη διεργασία χρήστη που δημιουργείται στο σύστημα
  - όλες οι άλλες δημιουργούνται από αυτή

Κλήσεις συστήματος

- Οι διεργασίες χρήστη αποκτούν πρόσβαση σε διάφορους πόρους μέσω των κλήσεων συστήματος
- Κάθε κλήση συστήματος μετατρέπεται σε μήνυμα το οποίο αποστέλεται στην κατάλληλη διεργασία διακομιστή για εξυπηρέτηση

Δομή του Πηγαίου Κώδικα του Minix (1)

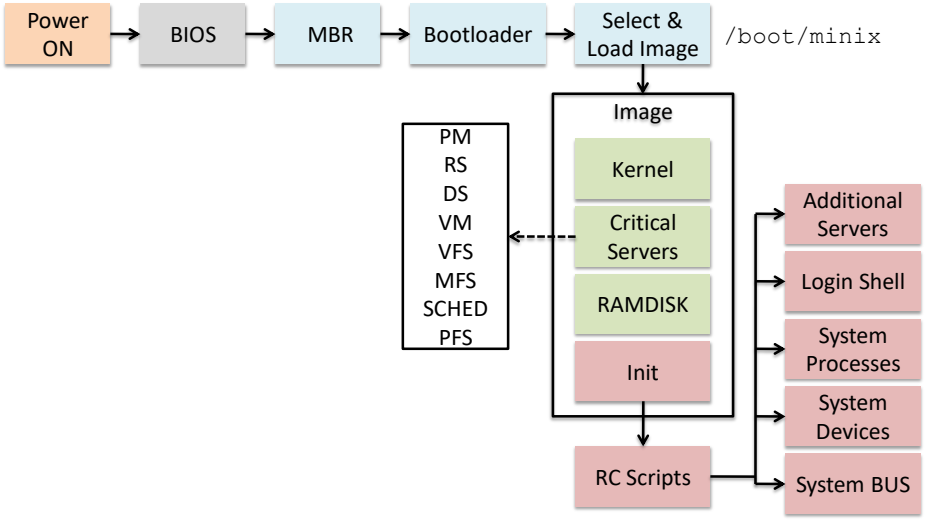
/usr/src/ - Πηγαίος κώδικας του Minix

- include/ - Περιλαμβάνει σημαντικά αρχεία κεφαλίδας
  - minix/ - Δηλώσεις που χρησιμοποιούνται από το Minix
- kernel/ - Ο κώδικας του πυρήνα του Minix. Περιέχει τον κώδικα που υλοποιεί το μηχανισμό περάσματος μηνυμάτων, την υλοποίηση διεργασιών, την υλοποίηση του ρολογιού του συστήματος κ.α
- servers/ - Περιέχει τον κώδικα για βασικούς διακομιστές
  - pm/ - Κώδικας διαχειριστή διεργασιών
  - vfs/ - Κώδικας συστήματος αρχείων
  - sched/ - Κώδικας χρονοδρομολογητή
  - inet/ - Κώδικας υποστήριξης δικτύωσης

Δομή του Πηγαίου Κώδικα του Minix (2)

- lib/ - Πηγαίος Κώδικας βιβλιοθηκών συστήματος (libc.a)
  libsys/ - Επικοινωνία πυρήνα με τις διεργασίες διακομιστή
- tools/ - Περιέχει διάφορα εργαλεία για το χτίσιμο του Minix
- boot/ - Κώδικας για την εκκίνηση του Minix
  minix/ - Εικόνες εκκίνησης
- bin/ και sbin/ - Εργαλεία που μπορούν να εκτελεστούν στο Minix

Εκκίνηση του Minix



Διαδιεργασιακή Επικοινωνία

Το Minix αποτελείται από ανεξάρτητα προγράμματα που επικοινωνούν μεταξύ τους μέσω μηνυμάτων

Αποστολή και λήψη μηνυμάτων

- `send(dest, &message)`
  αποστολή μηνύματος message στην διεργασία dest
- `receive(source, &message)`
  λήψη μηνύματος message από την διεργασία source
- `send_rec(src_dst, &message)`
  αποστολή μηνύματος message και αναμονή απάντησης από τη διεργασία src_dst

Δομή Μηνυμάτων (1)

Ορισμός της δομής των μηνυμάτων: `/usr/src/include/minix/ipc.h`

Δομές που περιέχουν την ένωση 6 διαφορετικών τύπων μηνυμάτων:

```
typedef struct {int m1i1, m1i2, m1i3; char *m1p1, *m1p2, *m1p3;} mess_1;
typedef struct {int m2i1, m2i2, m2i3; long m2l1, m2l2; char *m2p1;} mess_2;
typedef struct {int m3i1, m3i2; char *m3p1; char m3ca1[M3_STRING];} mess_3;
typedef struct {long m4l1, m4l2, m4l3, m4l4, m4l5;} mess_4;
typedef struct {short m5c1, m5c2; int m5i1, m5i2; long m5l1, m5l2, m5l3;}mess_5;
typedef struct {int m7i1, m7i2, m7i3, m7i4; char *m7p1, *m7p2;} mess_7;
typedef struct {int m8i1, m8i2; char *m8p1, *m8p2, *m8p3, *m8p4;} mess_8;
typedef struct {long m9l1, m9l2, m9l3, m9l4, m9l5;short m9s1, m9s2, m9s3, m9s4; } mess_9;
typedef struct {int m10i1, m10i2, m10i3, m10i4; long m10l1, m10l2, m10l3; } mess_10;

typedef struct {
    int m_source; // who sent the message
    int m_type; // what kind of message is it
    union {
        mess_1 m_m1;
        mess_2 m_m2;
        mess_3 m_m3;
        mess_4 m_m4;
        mess_5 m_m5;
        mess_6 m_m6;
        mess_7 m_m7;
        mess_8 m_m8;
        mess_9 m_m9;
        mess_10 m_m10;
    } m_u;
} message;
```


Δομή Μηνυμάτων (2)

Οι 2 βασικότεροι τύποι:

m_source	m_type	m1_i1	m1_i2	m1_i3	m1_p1	m1_p2	m1_p3

m_source	m_type	m2_i1	m2_i2	m2_i3	m2_l1	m2_l2	m2_p1

⋮

- `m_source` – αποστολέας μηνύματος
- `m_type` – τύπος μηνύματος
- το i υποδηλώνει ακέραιο, το p δείκτη, το l long, και το s short

Έστω ένα μήνυμα x, μπορούμε να χρησιμοποιήσουμε τον πρώτο τύπο μηνύματος και να προσπελάσουμε τον πρώτο ακέραιο ως εξής:

- `x.m_u.m_m1.m1i1`
- ή ακόμη πιο απλά `x.m1_i1`

Δρομολόγηση Διεργασιών (1)

Η χρονοδρομολόγηση μιας διεργασίας γίνεται είτε μόνο από τον πυρήνα, είτε από τον πυρήνα σε συνεργασία με το διακομιστή χρονοδρομολόγησης

Ουρές προτεραιότητας

- Πολλαπλές ουρές διαφορετικής προτεραιότητας
- Περιλαμβάνουν διεργασίες προς εκτέλεση
- Μια διεργασία μπορεί να βρίσκεται μόνο σε μια ουρά

Χρονοδρομολόγηση μόνο από τον πυρήνα

- Μόνο κρίσιμες διεργασίες συστήματος (και διεργασία του διακομιστή χρονοδρομολόγησης)
- Μπορεί να αναλάβει τη δρομολόγηση όλων των υπόλοιπων διεργασιών σε περίπτωση που ο διακομιστής χρονοδρομολόγησης δεν είναι διαθέσιμος
- Καθορισμός ουράς στην οποία θα εισαχθεί μια διεργασία
- Επιλογή επόμενης διεργασίας προς εκτέλεση

Δρομολόγηση Διεργασιών (2)

Χρονοδρομολόγηση με συνεργασία του πυρήνα και του διακομιστής χρονοδρομολόγησης

- Όλες οι διεργασίες για τις οποίες δεν είναι υπεύθυνος μόνο ο χρονοδρομολογητής του πυρήνα
- Ο διακομιστής χρονοδρομολόγησης καθορίζει την προτεραιότητα των διεργασιών
- Για κάθε διεργασία ζητάει από τον πυρήνα ώστε να την εισάγει στην επιλεγμένη ουρά με βάση μια προτεραιότητα που έχει υπολογίσει
- Η επιλογή της επόμενης προς εκτέλεση διεργασίας γίνεται από τον πυρήνα ο οποίος σέβεται τις προτεραιότητες που έχει καθορίσει ο διακομιστής χρονοδρομολόγησης

Πίνακες Διεργασιών

Πίνακας διεργασιών πυρήνα

- Τα στοιχεία του πίνακα διεργασιών πυρήνα έχουν τύπο `struct proc`
- Ορίζονται στο αρχείο `kernel/proc.h`

Πίνακας διεργασιών διαχειριστή διεργασιών

- Τα στοιχεία του πίνακα διεργασιών του διαχειριστή διεργασιών έχουν τύπο `struct mproc`
- Ορίζονται στο αρχείο `servers/pm/mproc.h`

Πίνακας διεργασιών διακομιστή χρονοδρομολόγησης

- Τα στοιχεία του πίνακα διεργασιών του διακομιστή χρονοδρομολόγησης έχουν τύπο `struct schedproc`
- Ορίζονται στο αρχείο `servers/sched/schedproc.h`

Χρονοδρομολόγηση Πυρήνα

Ο πυρήνας συντηρεί ελάχιστη πληροφορία χρονοδρομολόγησης για κάθε διεργασία

- Ουρά προτεραιότητας
- Εναπομείναντα χρόνο
- Διακομιστή χρονοδρομολόγησης που είναι υπεύθυνος για τη διεργασία

Πολιτική χρονοδρομολόγησης

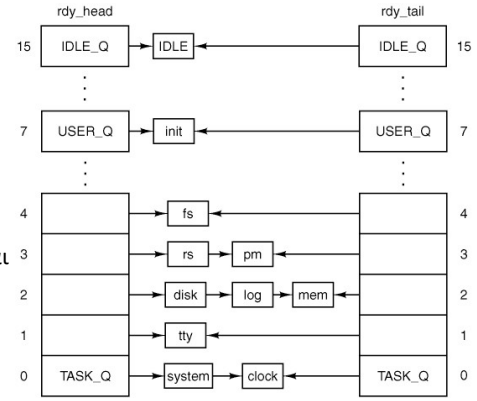
- Πολυεπίπεδη ανάδραση
- Χρησιμοποιείται για μια διεργασία μόνο όταν το πεδίο που δείχνει τον διακομιστή χρονοδρομολόγησης (`p_scheduler`) είναι NULL

Πολυεπίπεδη Ανάδραση (1)

Χρησιμοποιούνται 16 ουρές διεργασιών:

- αδρανής διεργασία (IDLE)
  - χαμηλότερη προτεραιότητα
- διεργασίες χρήστη (επίπεδο 4)
- διεργασίες διακομιστή (επίπεδο 3)
- οδηγοί συσκευών (επίπεδο 2)
- στοιχειώδεις εργασίες ρολογιού και συστήματος (επίπεδο 1)
  - υψηλότερη προτεραιότητα

Οι πίνακες `rdy_head` και `rdy_tail` περιέχουν το πρώτο και το τελευταίο στοιχείο, αντίστοιχα, κάθε ουράς



Πολυεπίπεδη Ανάδραση (2)

Ο αλγόριθμος λειτουργεί ως εξής:

- εντοπίζει την ουρά υψηλότερης προτεραιότητας που δεν είναι άδεια
- επιλέγει τη διεργασία που βρίσκεται στην αρχή της ουράς

Σε περίπτωση που όλες οι ουρές είναι άδειες εκτελείται η αδρανής διεργασία

Λήψη απόφασης δρομολόγησης:

1. Κατά την είσοδο/έξοδο διεργασίας από αποκλεισμό
2. Όταν ξεπεραστεί το κβάντο της εκτελούμενης διεργασίας

Πολυεπίπεδη Ανάδραση (3)

Στις ουρές περιέχονται μόνο οι διεργασίες που είναι έτοιμες προς εκτέλεση

Όταν μία διεργασία που βρισκόταν σε αποκλεισμό ξυπνήσει, προστίθεται στο τέλος της αντίστοιχης ουράς

Όταν μία διεργασία τερματίσει ή μπλοκάρει, απομακρύνεται από την αντίστοιχη ουρά

Οδηγός Ρολογιού στο Minix

Το σύστημα λαμβάνει HZ=60 διακοπές ρολογιού (ticks) το δευτερόλεπτο

Τις διακοπές ρολογιού χειρίζεται η συνάρτηση `timer_int_handler()` στο `kernel/clock.c`

- Ο **χρόνος χρήστη** χρεώνεται στη διεργασία χρήστη που έτρεχε όταν έγινε η διακοπή ρολογιού
- Ο **χρόνος συστήματος** χρεώνεται σε κάποια διεργασία όταν τρέχει εκ μέρους της κάποιος διακομιστής, πχ, το σύστημα αρχείων ή ο διαχειριστής μνήμης

Κώδικας Δρομολόγησης Πυρήνα

Ο αλγόριθμος δρομολόγησης υλοποιείται από την συνάρτηση `pick_proc()` στο αρχείο `src/kernel/proc.c`

Η `pick_proc()` **επιλέγει** την επόμενη προς εκτέλεση διεργασία και την επιστρέφει

- `bill_ptr`: δείχνει στη διεργασία χρήστη που θα χρεωθεί τον χρόνο συστήματος

Η `switch_to_user()` κάνει την αλλαγή επιπέδου από πυρήνα σε χρήστη για την επιλεγμένη διεργασία

- Ενημερώνεται η `proc_ptr`
- Η διεργασία πλέον χρονοδρομολογείται άπό τον διακομιστή χρονοδρομολόγησης

Διακομιστής Χρονοδρομολόγησης (1)

Ο διακομιστής χρονοδρομολόγησης υλοποιείται κάτω από το φάκελο `servers/sched/`

- Εκτελείται ως διεργασία
- Η χρονοδρομολόγησή του γίνεται από τον πυρήνα

Εφαρμογή πολιτικής χρονοδρομολόγησης

- Τρέχουσα πολιτική: Πολυεπίπεδη ανάδραση
- Καθορίζεται η ουρά στην οποία θα εισαχθεί κάθε διεργασία

Επιλογή επόμενης διεργασίας προς εκτέλεση

- Συνεχίζει να υλοποιείται στον πυρήνα στην `pick_proc()`

Διακομιστής Χρονοδρομολόγησης (2)

Ανάληψη χρονοδρομολόγησης μιας νέας διεργασίας

- Όταν μια διεργασία δημιουργείται με κλήση της `fork()`, η `do_fork()` του `forkexit.c` (`servers/pm/`) εισάγει τη νέα διεργασία στον πίνακα διεργασιών του διακομιστή διεργασιών
- Στη συνέχεια ο PM αιτείται από τον SCHED να αναλάβει τη χρονοδρομολόγηση της διεργασίας

Εισαγωγή νέας διεργασίας στον πίνακα διεργασιών

- `do_start_scheduling()` στο `servers/sched/schedule.c`
- Εισαγωγή της διεργασίας στον πίνακα `schedproc`
- Ενημέρωση πεδίων της διεργασίας
- Αίτημα στον πυρήνα ώστε να τη δρομολογήσει (να την εισάγει σε κάποια ουρά που έχει επιλεγεί από τον διακομιστή χρονοδρομολόγησης)

Διακομιστής Χρονοδρομολόγησης (3)

Δρομολόγηση διεργασίας

- `schedule_process()` στο `servers/sched/schedule.c`
- Θέτει την προτεραιότητα της διεργασίας (αριθμός ουράς), το κβάντο, και τους επεξεργαστές που μπορεί να εκτελεσθεί
- Στέλνει μήνυμα στον πυρήνα ώστε να εισάγει τη διεργασία στην επιλεγμένη ουρά (`sys_schedule()`). Αν η διεργασία βρίσκεται ήδη σε κάποια ουρά, τότε εξάγεται από αυτήν και εισάγεται στη νέα

Λήξη κβάντου διεργασίας

- Ο διακομιστής λαμβάνει το μήνυμα SCHEDULING_NO_QUANTUM στο `servers/sched/main.c`
- Καλείται η `do_noquantum()` στο `servers/sched/schedule.c`
- Ο διακομιστής αυξάνει το πεδίο προτεραιότητας κατά 1 για να τη βάλει σε ουρά χαμηλότερης προτεραιότητας
- Δρομολογεί εκ νέου η διεργασία με κλήση της `sys_schedule()`

Διακομιστής Χρονοδρομολόγησης (4)

Ενημέρωση προτεραιότητας

- `balance_queues()` στο `servers/sched/schedule.c`
- Πεδιοδικά ο διακομιστής διατρέχει τον πίνακα διεργασιών και τοποθετεί κάθε διεργασία σε ουρά υψηλότερης προτεραιότητας. Στη συνέχεια δρομολογεί ξανά την κάθε διεργασία

Επιλογή διεργασίας για εκτέλεση

- Η επιλογή της επόμενης διεργασίας προς εκτέλεση συνεχίζει να γίνεται από τον πυρήνα στην `pick_proc()`

2η Εργαστηριακή Άσκηση

Ορισμός Δίκαιης Δρομολόγησης

Ζητούμενο:

- Διαχωρισμός των διεργασιών χρήστη σε **ομάδες** και **ισομοιρασμός** του χρόνου επεξεργαστή ανάμεσα στις ομάδες

Έστω ότι έχουμε 2 ομάδες A,B και έστω ότι υπάρχουν οι ακόλουθες διεργασίες:

- στην A ομάδα η διεργασία A1 και
- στη B ομάδα οι διεργασίες B1,B2

Ο δίκαιος δρομολογητής αναθέτει:

- 50% του συνολικού χρόνου του στην ομάδα A και
- 50% του συνολικού χρόνου του στην ομάδα B

→ Άρα ο χρόνος του επεξεργαστή να διαμοιραστεί ως εξής:

- 50% στη διεργασία A1,
- 25% στη διεργασία B1 και
- 25% στη διεργασία B2

Οδηγός Ομάδας

Η ταυτότητα ομάδας μιας διεργασίας ορίζεται από την ταυτότητα της πρώτης διεργασίας της ομάδας

- πεδίο `mp_procgrp` στο `struct mproc`

Θα πρέπει να μεταφέρετε την ταυτότητα ομάδας από τη διεργασία διακομιστή (PM) στον διακομιστή χρονοδρομολόγησης (SCHED)

- Μπορείτε να συμπεριλάβετε την ταυτότητα της ομάδας της νέας διεργασίας στο μήνυμα που στέλνεται από τον PM στον SCHED
- Χρησιμοποιήστε κάποιο πεδίο του μηνύματος που δε χρησιμοποιείται ήδη

Πίνακας Δρομολόγησης

Τροποποιήστε τον πίνακα δρομολόγησης του διακομιστή χρονοδρομολόγησης ώστε κάθε θέση του πίνακα να περιλαμβάνει τα εξής πεδία:

- `procgrp`: Οδηγός ομάδας
- `proc_usage`: Χρήση διεργασίας
- `grp_usage`: Χρήση group διεργασιών
- `fss_priority`: Προτεραιότητα με βάση τον αλγόριθμο δίκαιης χρονοδρομολόγησης

Αρχικοποίηση πεδίων

- Για κάθε νέα διεργασία θα πρέπει να αρχικοποιείτε τα παραπάνω πεδία κατάλληλα

Ενημέρωση πεδίων

- Ενημερώστε τα παραπάνω πεδία σύμφωνα με τον αλγόριθμο δίκαιης χρονοδρομολόγησης όταν μια διεργασία ολοκληρώνει ένα κβάντο

Ουρές Δρομολόγησης

Στο `src/include/minix/config.h` θα πρέπει να μειώσετε το συνολικό πλήθος ουρών έτσι ώστε οι διεργασίες χρήστη να εισάγονται μόνο σε μια ουρά (`USER_Q`)

- `USER_Q`: Προεπιλεγμένη προτεραιότητα (ουρά) διεργασιών χρήστη
- `MAX_USER_Q`: Υψηλότερη προτεραιότητα διεργασιών χρήστη
- `MIN_USER_Q`: Χαμηλότερη προτεραιότητα διεργασιών χρήστη

Επαλήθευση Σωστής Εκτέλεσης

Η ομάδα (το πεδίο procgrp) είναι κοινή για όλες τις διεργασίες που εκτελούνται στο ίδιο session

Ο οδηγός μίας ομάδας είναι η πρώτη διεργασία που δημιουργείται στην ομάδα

Όταν ολοκληρώσετε όλα τα βήματα της άσκησης, μπορείτε να εξετάσετε αν ο αλγόριθμος δρομολόγησης εκτελείται σωστά με τη διαδικασία που περιγράφεται στο ερώτημα 4. Μπορείτε να χρησιμοποιήσετε και την εντολή top ώστε να πάρετε πληροφορίες για τις διεργασίες που εκτελούνται στο σύστημα (παρατηρήστε τη σειρά των διεργασιών που τρέχουν τα script σας)

Παράρτημα

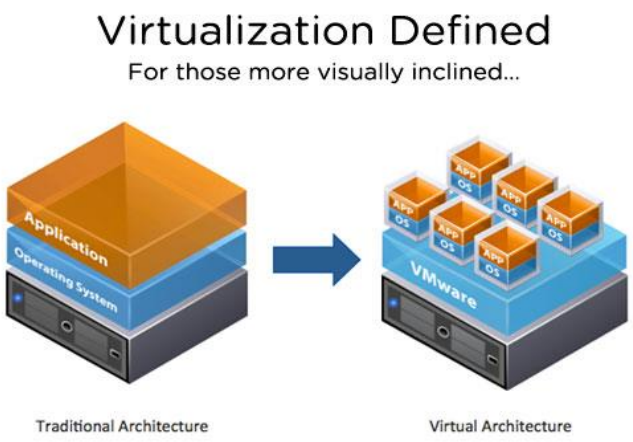
Περιβάλλον Εικονικοποίησης

Στην εργασία αυτή θα δουλέψετε στο περιβάλλον εικονικοποίησης **VMware Workstation Player**

Το λογισμικό εικονικοποίησης επιτρέπει σε πολλαπλά λειτουργικά συστήματα να εκτελούνται ταυτόχρονα στο ίδιο φυσικό μηχάνημα

Το λογισμικό αυτό τυπικά χρησιμοποιεί ένα λειτουργικό ως ξενιστή (host) στο οποίο και εκτελείται, και το οποίο υποστηρίζει ένα οποιοδήποτε αριθμό από φιλοξενούμενα (guest) λειτουργικά

Εικονικοποίηση



Εγκατάσταση του Minix 3.2.0

Κατεβάστε το αρχείο Minix3.2.0.zip και αποσυμπιέστε σε δίσκο USB μνήμης flash

Ακολουθείστε τις οδηγίες της εκφώνησης για την εκκίνηση της εικονικής μηχανής που περιλαμβάνει το Minix

Μπορείτε να εξετάσετε την σωστή λειτουργία του Minix μέσω κάποιων test στον κατάλογο `/usr/src/test`

(συνδεθείτε ως χρήστης root)

```
cd /usr/src/test
make
./run
```

Συνδεθείτε ως χρήστης root

Για να κλείσετε το Minix εκτελέστε `shutdown`

Τροποποίηση του Minix 3.2.0

Τροποποίηση μόνο του Image (Πυρήνας, διακομιστές PM, SCHED,..., init)

- Αφού μπείτε στον κατάλογο `/usr/src/tools`, εκτελέστε `make install` για να χτίσετε το νέο σύστημα και να εγκαταστήσετε το νέο image

Τροποποίηση βιβλιοθηκών

- Αφού μπείτε στον κατάλογο `/usr/src/` εκτελέστε `make libraries`
- Στη συνέχεια εκτελέστε `make install` στο `/usr/src/tools/`

Τοποποίηση αρχείων κεφαλίδας στο `include/`

- Αφού μπείτε στον κατάλογο `/usr/src/` εκτελέστε `make includes`
- Στη συνέχεια εκτελέστε `make install` στο `/usr/src/tools/`

Εναλλακτικά

- Εκτελέστε `make world` στο `/usr/src` ώστε να μεταγλωττίσετε και να εγκαταστήσετε τα πάντα

Το νέο image εμφανίζεται στον κατάλογο `/boot/minix/` με τον τελευταίο αριθμό έκδοσης

Κατά την εκκίνηση φορτώνεται στη μνήμη το νεότερο image που έχει δημιουργηθεί

Επαναφορά Συστήματος

Σε περίπτωση που το νέο image που δημιουργήσατε έχει πρόβλημα ή αποτυγχάνει και προκειμένου να αναιρέσετε τις αλλαγές που κάνατε, μπορείτε να επιλέξετε να κανετε boot με κάποιο παλιότερο image από το menu

Εναλλακτικά, μπορείτε να επαναφέρετε κάποιο συγκεκριμένο image ως εξής:

- κατά την εκκίνηση πιέστε "**3**"
- εισάγετε τις εντολές:
  - `load_mods /boot/minix/3.2.0r#/mod*`
  - `multiboot /boot/minix/3.2.0r#/kernel rootdevname=c0d0p0s0`
  - Όπου # ο αριθμός του image που θέλετε να φορτώσετε

Τροποποίηση Αρχείων στο Minix

Διαθέσιμοι κειμενογράφοι:

- vi
- zile (emacs-like)*
- nano*
- Elvis*

*θα πρέπει να τον εγκαταστήσετε εσείς

Εγκατάσταση νέων πακέτων

Θέστε τον nameserver της Google: Εισάγετε τον nameserver 8.8.8.8 στο /etc/resolv.conf ως εξής:

- `echo "nameserver 8.8.8.8" > /etc/resolv.conf`

Ενημερώστε την τοπική βάση των πακέτων του Minix:

- `pkgin update`

Αναζητείστε ένα πακέτο:

- `pkgin search <packagename>`
- Ή `pkgin available` ώστε να πάρετε μια λίστα όλων των διαθέσιμων πακέτων

Εγκαταστείστε ένα νέο πακέτο:

- `pkgin install <packagename>`
- Πχ: `pkgin install zile`

Μεταφορά αρχείων

Μπορείτε να χρησιμοποιήσετε το scp ώστε να μεταφέρετε αρχεία από/προς το Minix και το ssh ώστε να συνδεθείτε στην εικονική μηχανή που τρέχει το Minix

- Προσοχή: Τα ssh/scp από/προς την εικονική μηχανή του Minix ΔΕ λειτουργούν στα PC του εργαστηρίου. Χρησιμοποιήστε τα μόνο σε δικό σας υπολογιστή!
- Βρείτε την τρέχουσα διεύθυνση IP εκτελώντας την εντολή ifconfig
- Χρησιμοποιήστε το scp για να μεταφέρετε κάποιο αρχείο από/πρός την παραπάνω διεύθυνση
  - Πχ:

    local-pc# scp myfile.txt 172.16.149.128:/usr/src/.

    local-pc# scp 172.16.149.128:/usr/src/myfile.txt .

turnin

Μόλις έρθει η στιγμή να παραδώσετε την άσκηση

- Χρησιμοποιήστε το scp για να πάρετε τα αρχεία που τροποποιήσατε και το τελευταίο image που αντιστοιχεί στα αρχεία αυτά. Δημιουργήστε το image.tar ως εξής:
  - `cd /root; tar cvf image.tar /boot/minix/3.2.0r#` (#: αριθμός image)
- Γράψτε το readme και μετατρέψτε το σε **PDF**
  - Συμπεριλάβετε στην αρχή ονόματα και αριθμούς μητρώου. Μην επαναλάβετε την εκφώνηση της άσκησης. Εξηγήστε τη λύση σας και δικαιολογήστε γιατί δουλεύει συμπεριλαμβάνοντας παραδείγματα που δοκιμάσατε, την έξοδο που πήρατε και παρατηρήσεις
- Θα πρέπει να έχετε τα εξής αρχεία:
  - readme.pdf
  - image.tar
  - file1.c file2.c file3.h …
- Μη δημιουργήσετε συμπιεσμένο αρχείο που θα περιλαμβάνει τα παραπάνω αρχεία αλλά στείλτε όλα τα αρχεία ξεχωριστά:
  - turnin lab2_16@myy601 readme.pdf image.tar file1.c file2.c …

Κατά την εξέταση

- Να έχετε μαζί σας την εικονική μηχανή στην οποία δουλέψατε σε USB flash memory (ή στο λογαριασμό σας στο τμήμα)

Χρήσιμοι σύνδεσμοι

Ο πλήρης κώδικας του Minix 3.2.0 είναι διαθέσιμος στο link:
http://www.cs.uoi.gr/~stergios/teaching/myy601/lab/minix/html/

Επίσης μπορείτε να βρείτε χρήσιμο υλικό στο παρακάτω site:
http://wiki.minix3.org/doku.php?id=releases:3.2.0:start

??...Απορίες...??

Ώρες γραφείου

- Κάθε **Δευτέρα 12:00 – 13:00** στο γραφείο Α34

FORUM Μαθήματος

- Όλες τις υπόλοιπες μέρες και ώρες
- http://ecourse.uoi.gr/course/view.php?id=44