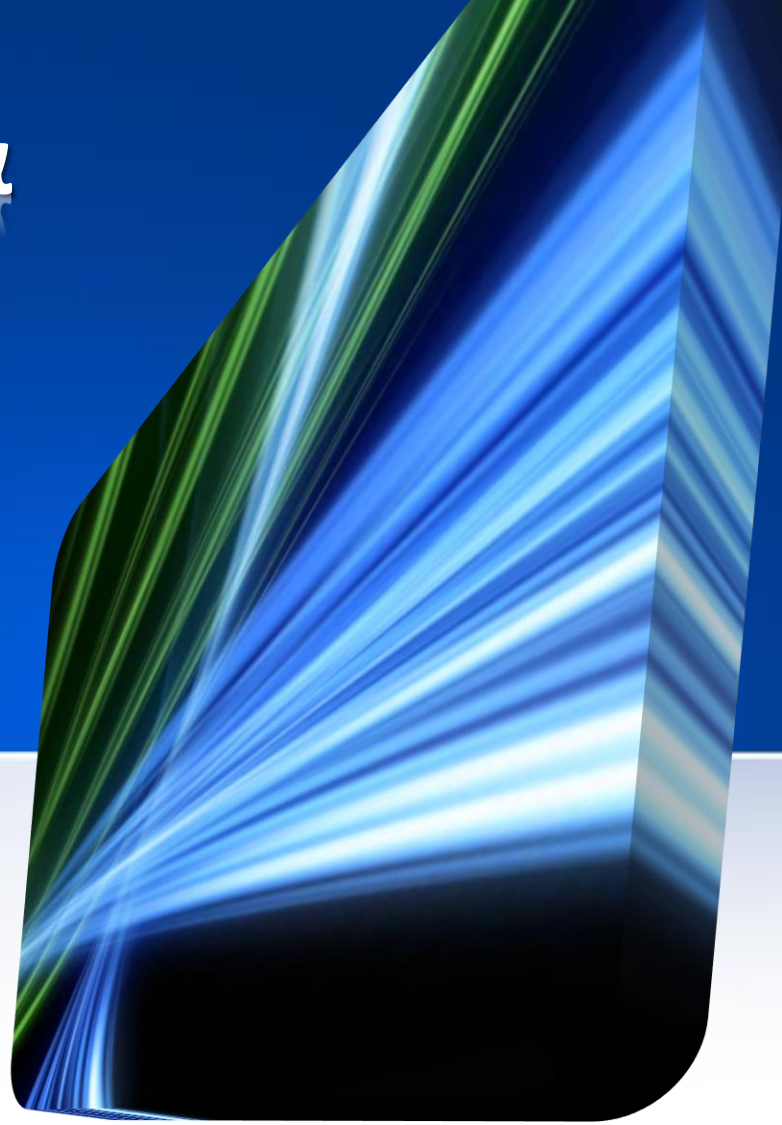
Λειτουργικά Συστήματα
6ο εξάμηνο ΣΗΜΜΥ
Ακ. έτος 2019-2020



Εργαστηριακή Άσκηση 2

Θεωρία Εργ. Άσκησης 2
Σήματα

Σήμα (signal) είναι ένας τρόπος επικοινωνίας των διεργασιών.

Είναι μια ειδοποίηση που στέλνεται σε μια διεργασία προκειμένου να την ενημερώσει για κάποιο **γεγονός**.

Η διεργασία που λαμβάνει ένα σήμα **διακόπτει** την εκτέλεση της και αναγκάζεται να χειριστεί το σήμα άμεσα.

Θεωρία Εργ. Άσκησης 2
Σήματα

Κάθε σήμα έχει έναν ακέραιο αριθμό που το αντιπροσωπεύει (1,2, …), καθώς επίσης και ένα συμβολικό όνομα που καθορίζεται συνήθως στο αρχείο /usr/include/signal.h

Με την εντολή "**kill -l**" βλέπουμε τον κατάλογο σημάτων που υποστηρίζονται από το σύστημά μας.

Θεωρία Εργ. Άσκησης 2
Λίστα Σημάτων

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at terminal
SIGTTIN	21,21,26	Stop	Terminal input for background process
SIGTTOU	22,22,27	Stop	Terminal output for background process

Θεωρία Εργ. Άσκησης 2
Σήματα

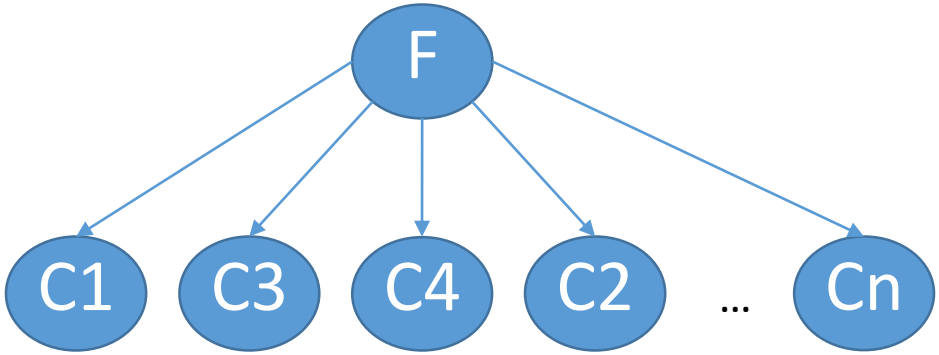
Κάθε σήμα μπορεί να έχει έναν **χειριστή σήματος** (handler), ο οποίος είναι μια συνάρτηση που εκτελείται όταν η διεργασία λαμβάνει το συγκεκριμένο σήμα. Η συνάρτηση αυτή καλείται ασύγχρονα.

Όταν στέλνεται το σήμα σε μια διεργασία, το ΛΣ σταματά την εκτέλεση της διεργασίας, και "την αναγκάζει" να καλέσει την αντίστοιχη συνάρτηση χειρισμού αυτού του σήματος (handler).

Όταν η συνάρτηση ολοκληρώσει την εκτέλεσή της, η διεργασία συνεχίζει την εκτέλεση από το σημείο που βρισκόταν αμέσως πριν παραληφθεί το σήμα.

Εργαστηριακή Άσκηση 2

Να γραφτεί πρόγραμμα C σε περιβάλλον Linux, στο οποίο η διεργασία-πατέρας (F) δημιουργεί **n** διεργασίες-παιδιά (C1, C2, C3, C4 … Cn) σύμφωνα με το παρακάτω δέντρο διεργασιών:



Ο χρήστης θα μπορεί να στέλνει σήματα SIGUSR1, SIGUSR2, SIGTERM στην διεργασία πατέρα και στις διεργασίες παιδιά. Ο χειρισμός αυτών των σημάτων θα εξηγηθεί αναλυτικά στις επόμενες διαφάνειες

Εργαστηριακή Άσκηση 2

Παράδειγμα εκτέλεσης προγράμματος

./ask2 d1 d2 d3 … dn

Τα ορίσματα **d1**, **d2**, **d3 … dn** πρέπει να είναι ακέραιοι θετικοί αριθμοί που προσδιορίζουν μια καθυστέρηση σε δευτερόλεπτα. Το όρισμα **di** θα χρησιμοποιηθεί αντίστοιχα από την διεργασία Ci (*δες διαφάνεια 12*).

Το πλήθος των ορισμάτων di προσδιορίζει το πλήθος των διεργασιών (**n**) που θα δημιουργήσει ο πατέρας, δηλαδή **n** = argc -1

Εργαστηριακή Άσκηση 2

Όλες οι διεργασίες παιδιά μόλις δημιουργηθούν εκτυπώνουν το μήνυμα:

[Child Process x: y] Was created and will pause!

x είναι η αρίθμηση της διεργασίας ($1 \leq \mathbf{x} \leq \mathrm{n}$) και

y είναι το **pid** της

π.χ. **[Child Process 2: 3247] Was created and will pause!**

Οι διεργασίες παιδιά αφού εκτυπώσουν το μήνυμα αναστέλλουν την εκτέλεση τους (κατάσταση pause)

Εργαστηριακή Άσκηση 2

Μια διεργασία μπορεί να βρεθεί σε κατάσταση pause

- Είτε καλώντας την συνάρτηση **pause()**, η οποία αναστέλλει την εκτέλεση της διεργασίας που την καλεί μέχρι να λάβει ένα σήμα
- Είτε στέλνοντας στον εαυτό της το σήμα SIGSTOP με τη συνάρτηση **raise()**, δηλαδή εκτελώντας **raise(SIGSTOP)**

Όταν στέλνεται το σήμα SIGSTOP σε μια διεργασία, τότε προκαλείται η παύση της διεργασίας αυτής. Η διεργασία θα ξαναρχίσει την εκτέλεση της μόνο εάν της αποσταλεί το σήμα SIGCONT

Εργαστηριακή Άσκηση 2

Στην συνέχεια η διεργασία πατέρας εκκινεί όλες τις διεργασίες παιδιά με την σειρά, στέλνοντάς τους το σήμα SIGCONT

Μια διεργασία μπορεί να στείλει ένα σήμα σε μια άλλη, με την κλήση **kill(pid, SIGNAL)** η οποία στέλνει το σήμα με όνομα **SIGNAL** στη διεργασία με process id ίσο με το όρισμα **pid**.

Για παράδειγμα με την εκτέλεση της εντολή **kill(3247, SIGCONT)** στέλνεται το σήμα SIGCONT στην διεργασία με pid = 3247

Εργαστηριακή Άσκηση 2

Πριν η διεργασία πατέρας εκκινήσει την διαδικασία αποστολής των σημάτων SIGCONT στις διεργασίες παιδιά μπορεί να τις περιμένει να αναστείλουν όλες την λειτουργία τους (pause) με την χρήση της **waitpid()**

Η **waitpid()** χρησιμοποιείται για να περιμένει η καλούσα διεργασία αλλαγές κατάστασης σε μια διεργασία παιδί της και να λάβει πληροφορίες σχετικά με την διεργασία της οποίας η κατάσταση έχει αλλάξει

Εργαστηριακή Άσκηση 2

Ορισμός **waitpid()**

pid_t waitpid(pid_t pid, int *wstatus, int options)

- Το πεδίο **pid** ορίζει το pid της διεργασίας που περιμένουμε να αλλάξει κατάσταση (με τιμή ίση με -1 περιμένουμε οποιαδήποτε διεργασία παιδί)
- Το πεδίο **wstatus** είναι ο δείκτης στην μεταβλητή στην οποία μπορούμε να αποθηκεύσουμε την κατάσταση της διεργασίας παιδί
- Το πεδίο **options** ορίζει επιπλέον επιλογές

Περισσότερες πληροφορίες
http://www.man7.org/linux/man-pages/man2/waitpid.2.html

Εργαστηριακή Άσκηση 2

Όλες οι διεργασίες μόλις βγουν από την κατάσταση pause εκτυπώνουν το εξής μήνυμα

[Child Process x: y] Is starting!

και εκτελούν την κύρια εργασία τους που είναι η επαναλαμβανόμενη αύξηση ενός μετρητή (**cnt**) με συγκεκριμένη καθυστέρηση που έχει δοθεί από τον χρήση ως όρισμα (di) κατά την εκτέλεση του προγράμματος (*δες διαφάνεια 7*). Για παράδειγμα η διεργασία παιδί C3 θα εκτελέσει τα παρακάτω

```
cnt = 0 ;
while(1) {
        cnt ++;
        sleep(d3);
}
```

Σημείωση: κάθε διεργασία χειρίζεται τον δικό της μετρητή **cnt**.

Εργαστηριακή Άσκηση 2

Όσο η διεργασίες-παιδιά εκτελούνται, ο χρήστης μπορεί να τους στείλει από terminal τα παρακάτω σήματα

SIGUSR1, SIGUSR2 ή SIGTERM

Η εντολή αποστολής ενός σήματος από terminal είναι

kill –SIGNAL pid

όπου **SIGNAL** το όνομα του σήματος που θέλουμε να αποσταλεί και **pid** το process id της διεργασίας στην οποία θέλουμε να στείλουμε το σήμα. Για παράδειγμα η εντολή **kill –SIGUSR1 3216** στέλνει το σήμα **SIGUSR1** στην διεργασία με pid = 3216

Εργαστηριακή Άσκηση 2

- Όταν μια διεργασία-παιδί λάβει το σήμα SIGUSR1 εκτυπώνει το μήνυμα **[Child Process x: y] Value: z! ,** όπου z η τρέχουσα τιμή του μετρητή της **cnt**

- Όταν μια διεργασίες παιδί λάβει το σήμα SIGUSR2 εκτυπώνει το μήνυμα **[Child Process y] Echo!**

- Όταν μια διεργασίες παιδί λάβει το σήμα SIGTERM τερματίζει την εκτέλεση της

Σημείωση: x είναι η αρίθμηση της διεργασίας και **y** είναι το pid της

Εργαστηριακή Άσκηση 2

Όσο η **διεργασία-πατέρας** εκτελείται, ο χρήστης μπορεί να του στείλει από terminal τα παρακάτω σήματα

SIGUSR1, SIGUSR2 ή SIGTERM

- Όταν η διεργασία πατέρας λάβει το σήμα SIGUSR1 εκτυπώνει το μήνυμα **[Father process: y] Will ask current values (SIGUSR1) from all active children processes.**

  και στέλνει σε όλες τις διεργασίες παιδιά το σήμα SIGUSR1, με αποτέλεσμα όλες οι διεργασίες παιδία να εκτυπώσουν το μήνυμα **[Child Process x: y] Value: z! ,** όπου z η τρέχουσα τιμή του μετρητή τους **cnt**

Σημείωση: x είναι η αρίθμηση της διεργασίας και **y** είναι το pid της

Εργαστηριακή Άσκηση 2

- Όταν η διεργασία-πατέρας λάβει το σήμα SIGUSR2 εκτυπώνει το μήνυμα **[Process y] Echo!**

- Όταν η διεργασία-πατέρας λάβει το σήμα SIGUTERM* τερματίζει κάθε διεργασία-παιδί αφού πρώτα εκτυπώσει το μήνυμα

  [Father process: y] Will terminate (SIGTERM) child process x: y

Σημείωση: x είναι η αρίθμηση της διεργασίας και **y** είναι το pid της

* Η διεργασία πατέρας να έχει την ίδια συμπεριφορά στο σήμα SIGINT

Εργαστηριακή Άσκηση 2

Μια διεργασία μπορεί να ορίσει τον χειριστή (handler) κάθε σήματος που θα λάβει με τις συναρτήσεις:

- **sigaction()** (προτείνεται)
- **signal()**

Η κλήση **signal(SIGNAL, handler)** καθορίζει ότι όταν η καλούσα διεργασία λάβει σήμα με όνομα **SIGNAL** θα εκτελέσει τη συνάρτηση χειρισμού σήματος **handler**.

Εργαστηριακή Άσκηση 2

Η κλήση συστήματος sigaction () χρησιμοποιείται για να αλλάξει τη δράση (action) που θα γίνει από μια διεργασία κατά την λήψη συγκεκριμένου σήματος.

int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);

- Η παράμετρος **signum** καθορίζει το σήμα και μπορεί να είναι οποιοδήποτε έγκυρο σήμα εκτός από SIGKILL και SIGSTOP
- Εάν η παράμετρος **act** είναι not NULL τότε θέτει το νέο sigaction για το δεδομένο signal
- Εάν η παράμετρος **oldact** είναι not NULL τότε εκεί αποθηκεύεται η προηγούμενη ορισμένη δράση για το δεδομένο signal

Εργαστηριακή Άσκηση 2

Η παράμετρος **act** και **oldact** είναι δομές τύπου **struct sigaction**

```
struct sigaction {
  void (*sa_handler)(int);                    // καθορίζει τη δράση που πρέπει να συσχετιστεί με το signum
  void (*sa_sigaction)(int, siginfo_t *, void *);   // Εάν ορίζεται SA_SIGINFO στο sa_flags, τότε
                                              // στο sa_sigaction (αντί του sa_handler) καθορίζεται η λειτουργία χειρισμού για το signum
                                              // μην εκχωρήσετε τιμή και  στα δύο:  sa_handler και sa_sigaction
  sigset_t sa_mask;                           //καθορίζει μια μάσκα σημάτων που πρέπει να αποκλειστούν
  int  sa_flags;                              //καθορίζει ένα σύνολο σημαιών που τροποποιούν τη συμπεριφορά του σήματος
  void     (*sa_restorer)(void);              // Το πεδίο δεν προορίζεται για application use

                                        };
```

Εργαστηριακή Άσκηση 2

Παράδειγμα βασικής χρήσης **sigaction()**

```
struct sigaction action;                    //δήλωση δομής struct sigaction
action.sa_handler =myfunction;              //ορισμός ονόματος συνάρτησης χειρισμού
sigaction(SIGUSR1, &action, NULL);          //ορισμός χειριστή (myfunction) για το σήμα SIGUSR1
```

Αποτέλεσμα: όταν η διεργασία λάβει το σήμα **SIGUSR1** θα διακόψει την εκτέλεση της και θα καλέσει την συνάρτηση **myfunction()**

Περισσότερες πληροφορίες

http://man7.org/linux/man-pages/man2/sigaction.2.html

Εργαστηριακή Άσκηση 2

Επιπλέον, μετά την πάροδο συγκεκριμένου χρόνου οι διεργασίες παιδιά τερματίζουν.

Για να το πετύχουν αυτό οι διεργασίες μπορούν να κάνουν κλήση της συνάρτησης **alarm()**

Η συνάρτηση **alarm(unsigned int seconds)** θα προκαλέσει το σύστημα να παράγει ένα σήμα SIGALRM, προς τη διεργασία που την κάλεσε, μετά την παρέλευση του αριθμού των δευτερόλεπτων πραγματικού χρόνου που καθορίζονται από την παράμετρο **seconds**.

Όταν οι διεργασίες τερματίσουν με αυτό το τρόπο θα εκτυπώσουν το μήνυμα **[Child process x: y] Time Expired! Final Value: z**

Ο χρόνος που έχετε επιλέξει εμφανίζεται στην αρχή της εκτέλεσης με το μήνυμα **Maximum execution time of children is set to 100 secs**

Εργαστηριακή Άσκηση 2

Να γίνεται έλεγχος λαθών σε κλήσεις συστήματος και στον χειρισμό σημάτων με κατάλληλη εκτύπωση.

Παράδειγμα εκτέλεσης 1

Αρχείο: first_example.mp4

Εκτέλεση: **./ask2 2 4 6 8**

Χρόνος εκτέλεσης: 100 secs

Πλήθος διεργασιών: 4

Επίδειξη σημάτων: SIGUSR1, SIGUSR2, SIGTERM

Παράδειγμα εκτέλεσης 2

Αρχείο: second_example.mp4

Εκτέλεση: **./ask2 1 4 24**

Χρόνος εκτέλεσης: 50 secs

Πλήθος διεργασιών: 3

Επίδειξη τερματισμού διεργασιών μετά το πέρας των 50 secs

Χρήσιμη αναφορά

http://man7.org/linux/man-pages/man7/signal.7.html