Εργαστήριο 5 – fork(), exec(), signals

Στο εργαστήριο θα μελετηθούν:

- *Παραδείγματα χρήσης των συναρτήσεων fork και exec*
  - *Συνάρτηση waitpid*
  - *Συνάρτηση* WIFEXITED
  - *Συνάρτηση* WEXITSTATUS
- *Παράδειγμα χρήσης σημάτων*

Διδάσκων: Γιώργος Χατζηπολλάς

Η συνάρτηση waitpid()

- int **wait** (**int** ***status**)
  - Προκαλεί την αναμονή μιας διεργασίας μέχρι κάποιο παιδί της τερματίσει
  - -1 αν η διεργασία δεν έχει παιδιά
- int **waitpid** (**pid_t** *pid*, **int** ***status**, **int** *options*)
  - Προκαλεί την αναμονή μιας διεργασίας μέχρι το παιδί της που καθορίζεται από το ***pid*** *να* τερματίσει
    - pid = -1 (ή WAIT_ANY) σημάνει ότι περιμένουμε πληροφορία για οποιοδήποτε παιδί
    - Δίνεται η δυνατότητα καθορισμού επιπλέον επιλογών. Η πιο συνηθισμένη επιλογή είναι η WNOHANG, η οποία εμποδίζει την waitpid να γίνει blocked αν δεν υπάρχουν τερματιζόμενα παιδιά
    - Το ***status*** μπορεί να τύχει επεξεργασίας με τα ακόλουθα macros τα οποία βρίσκονται στην βιβλιοθήκη wait.h
      - ***WIFEXITED(status)*** : Εάν δεν επιστρέψει 0 τότε το παιδί έχει τερματιστεί φυσιολογικά
      - **WEXITSTATUS(*status*):** Επιστρέφει τον κωδικό εξόδου (exit status) της διαδικασίας παιδί. Αυτό μπορεί να υπολογιστεί μόνο στην περίπτωση που η **WIFEXITED** επιστρέψει μη-μηδέν
      - **WIFSIGNALED(*status*):** Επιστρέφει κατά πόσο η διαδικασία παιδί έχει τερματίσει μη φυσιολογικά (λόγω κάποιου σήματος) ή επιστρέφει 0
      - **WTERMSIG(*status*):** Επιστρέφει το σήμα το οποίο ανάγκασε τη διαδικασία παιδί να τερματίσει. Αυτό μπορεί να υπολογιστεί μόνο στην περίπτωση που η **WIFSIGNALED** επιστρέψει μη-μηδέν

Παράδειγμα 1: Χρήση fork()

```
void fork1()
{
   pid_t pid[N];
   int i, child_status;

   for (i = 0; i < N; i++)
    if ((pid[i] = fork()) == 0) {
            sleep(20-2*i);
            exit(100+i);
    }
   for (i = 0; i < N; i++) {
    pid_t wpid = waitpid(pid[i], &child_status, 0);
    pid_t wpid = wait(&child_status);
    if (WIFEXITED(child_status))
       printf("Child %d terminated with exit status %d\n",
              wpid, WEXITSTATUS(child_status));
    else
       printf("Child %d terminate abnormally\n", wpid);
   }
}
```

- Συγχρονισμός με πολλά παιδιά
- Επεξεργασία των παιδιών σε τυχαία σειρά
- Χρήση της WIFEXITED και της WEXITSTATUS για να πάρουμε τις πληροφορίες από τα παιδιά

Παράδειγμα 2: Χρήση fork()

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[])
{  int i, depth, numb, pid1, pid2, status;
   if (argc > 1) depth = atoi(argv[1]);
   else          exit(0);
   if (depth > 5) {
      printf("Depth should be up to 5\n"); exit(0); }
   numb = 1;
   for(i=1 ; i<=depth ; i++) {
      printf("I am process no %2d with PID %5d and PPID %5d\n",
             numb, getpid(), getppid());
      switch (pid1 = fork()) {
         case 0:
            numb = 2*numb; break;
         case -1:
            perror("fork"); exit(1);
         default:
            switch (pid2 = fork()) {
               case 0:
                  numb = 2*numb+1; break;
               case -1:
                  perror("fork"); exit(1);
               default:
                  wait(&status); wait(&status);
                  exit(0); } } } }
```

Παράδειγμα 2
Παράδειγμα Εκτέλεσης

```
[hpollas@cs4090 epl428]$ tree 3
I am process no  1 with PID 11539 and PPID  9005
I am process no  2 with PID 11540 and PPID 11539
I am process no  4 with PID 11541 and PPID 11540
I am process no  5 with PID 11544 and PPID 11540
I am process no  3 with PID 11547 and PPID 11539
I am process no  6 with PID 11548 and PPID 11547
I am process no  7 with PID 11551 and PPID 11547
[hpollas@cs4090 epl428]$
```

```
[hpollas@cs4090 epl428]$ tree 4
I am process no  1 with PID 11486 and PPID  9005
I am process no  2 with PID 11487 and PPID 11486
I am process no  4 with PID 11488 and PPID 11487
I am process no  8 with PID 11489 and PPID 11488
I am process no  3 with PID 11491 and PPID 11486
I am process no  6 with PID 11493 and PPID 11491
I am process no 12 with PID 11494 and PPID 11493
I am process no  9 with PID 11495 and PPID 11488
I am process no 13 with PID 11501 and PPID 11493
I am process no  5 with PID 11500 and PPID 11487
I am process no 10 with PID 11502 and PPID 11500
I am process no  7 with PID 11504 and PPID 11491
I am process no 14 with PID 11506 and PPID 11504
I am process no 11 with PID 11509 and PPID 11500
I am process no 15 with PID 11511 and PPID 11504
[hpollas@cs4090 epl428]$
```

- Πρόγραμμα το οποίο δημιουργεί ένα δυαδικό δέντρο από διεργασίες το οποίο έχει δεδομένο βάθος N.
  - Κάθε διεργασία του δέντρου εκτυπώνει την ταυτότητά της, του πατέρα της καθώς και ένα αύξοντα αριθμό σύμφωνα με μία κατά πλάτος διάσχιση του δέντρου

Αποστολή/Παραλαβή Σημάτων
Άσκηση 1

```
#include <stdio.h>  /* For printf */
#include <stdlib.h> /* For exit */
#include <unistd.h> /* For execvp */
#include <signal.h> /* For SIGALRM, SIGKILL */

int pid;
void onalarm(int);


void onalarm(int sig)
{
  kill(pid, SIGKILL);
}
```

Αποστολή/Παραλαβή Σημάτων
Άσκηση 1

```
main(int argc, char *argv[])
{
  int sec = 10, status; char *progname;
  progname = argv[0];
  if (argc > 1 && *argv[1] == '-') {
        sec = atoi(argv[1]+1);
        argc--;
        argv++; }
  if (argc < 2) {
        printf("Usage: %s [-10] command\n", progname);
        exit(1); }
  if ((pid = fork()) < 0) {
        perror("fork"); exit(1); }
  if (pid == 0) {
        execvp(argv[1], &argv[1]);
        perror("execvp"); exit(1); }
  signal(SIGALRM, onalarm);
  alarm(sec);
  wait(&status);
  if ((status & 0377) == 0) { // if (WIFEXITED(status)) {
        printf("Command execution finished");
        printf(" with exit code %d\n", status >> 8); } // WEXITSTATUS(status))
  else
        printf("Command was killed by signal %d\n", status & 0177);
                                                    // WTERMSIG(status)
}
```

Αποστολή/Παραλαβή Σημάτων Άσκηση 1 Παράδειγμα Εκτέλεσης

```
$ ./timeout
Usage: ./timeout [-10] command
$ ./timeout -30
Usage: ./timeout [-10] command
$ ./timeout blabla
execvp: No such file or directory
Command execution finished with exit code 1
$ ./timeout -20 ps
PID TTY TIME CMD
12947 pts/0 00:00:00 bash
13026 pts/0 00:00:00 timeout
13027 pts/0 00:00:00 ps
Command execution finished with exit code 0
$ date ; ./timeout -12 sleep 15 ; date
Mon Feb 18 18:43:20 EET 2002
Command was killed by signal 9
Mon Feb 18 18:43:32 EET 2002
$ date ; ./timeout sleep 7 ; date
Mon Feb 18 18:43:49 EET 2002
Command execution finished with exit code 0
Mon Feb 18 18:43:56 EET 2002
$
```

Αποστολή/Παραλαβή Σημάτων
Άσκηση 2

```
int loop;
void sigusr1_handler(int);

main()
{
  int pid;
  signal(SIGUSR1, sigusr1_handler);

  while(1) {
        printf("I am process %d and I am going to loop\n", getpid());
        loop = 1;
        while(loop) {
                sleep(1); }
        if ((pid = fork()) == -1) {
                perror("fork"); exit(1); }
        if (pid == 0)
                continue;
        else
                exit(0); }
}

void sigusr1_handler(int sig)
{
  printf("SIGUSR1 (= %d) received\n", sig);
  loop = 0;
}
```

Διαχείριση Διεργασιών
Παράδειγμα Εκτέλεσης

```
$ ./persistent &
[1] 13059
I am process 13059 and I am going to loop
$ kill -USR1 13059
SIGUSR1 (= 10) received
I am process 13060 and I am going to loop
[1]+ Done                    ./persistent
$ kill -USR1 13060
$ SIGUSR1 (= 10) received
I am process 13061 and I am going to loop
kill -USR1 13061
SIGUSR1 (= 10) received
$ I am process 13062 and I am going to loop
kill -USR1 13062
$ SIGUSR1 (= 10) received
I am process 13063 and I am going to loop
kill -KILL 13063
$
```