ΕΡΓΑΣΤΗΡΙΟ 7

Σήματα

Να πληκτρολογηθεί και να εκτελεστεί ο κώδικας των επόμενων παραδειγμάτων που επιδεικνύουν τη χρήση των σημάτων ως μέσο επικοινωνία μεταξύ διεργασιών.

Παράδειγμα 1. Σύλληψη του σήματος SIGINT (αρχείο sigExample1.c).

```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
#include <stdlib.h>

void sig_handler(int signo) {
   if (signo == SIGINT)
      printf("received SIGINT\n"); }

int main(void) {
   if (signal(SIGINT, sig_handler) == SIG_ERR)
      printf("\ncan't catch SIGINT\n");
 // A long long wait so that we can easily
 // issue a signal to this process
 while(1)
    sleep(1);
 return 0; }
```

Παράδειγμα 2. Αδυναμία σύλληψης των σημάτων SIGKILL και SIGSTOP (αρχείο sigExample2.c).

```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
#include <stdlib.h>

void sig_handler(int signo) {
   if (signo == SIGKILL)
      printf("received SIGKILL\n");
   if (signo == SIGSTOP)
      printf("received SIGSTOP\n"); }

int main(void) {
   if (signal(SIGKILL, sig_handler) == SIG_ERR)
      printf("\ncan't catch SIGKILL\n");
   if (signal(SIGSTOP, sig_handler) == SIG_ERR)
      printf("\ncan't catch SIGSTOP\n");
 // A long long wait so that we can easily
 // issue a signal to this process
 while(1)
   sleep(1);
 return 0; }
```

Παράδειγμα 3. Τα σήματα γενικής χρήσεως SIGUSR1 και SIGUSR2 (αρχείο sigExample3.c).

```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
#include <stdlib.h>

void sig_handler(int signo) {
   if (signo == SIGUSR1)
      printf("received USR1\n");
   if (signo == SIGUSR2)
      printf("received USR2\n"); }

int main(void) {
   if (signal(SIGUSR1, sig_handler) == SIG_ERR)
       printf("\ncan't catch USR1\n");
   if (signal(SIGUSR2, sig_handler) == SIG_ERR)
       printf("\ncan't catch USR2\n");
 // A long long wait so that we can easily
 // issue a signal to this process
 while(1)
    sleep(1);
 return 0; }
```

Παράδειγμα 4. Ανταλλαγή σήματος μεταξύ γονικής και θυγατρικής διεργασίας – Παράδειγμα Α (αρχείο sigExample4.c).

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

void sighup() { // handler for SIGHUP
    signal(SIGHUP, sighup);
    printf("CHILD: I have received a SIGHUP\n"); }

void sigint() { // handler for SIGINT
    signal(SIGINT, sigint); /* reset signal */
    printf("CHILD: I have received a SIGINT\n"); }

void sigquit() { // handler for SIGQUIT
   printf("My DADDY has Killed me!!!\n");
   exit(0); }

void main() {
    int pid;
    signal(SIGHUP, sighup);
    signal(SIGINT, sigint);
    signal(SIGQUIT, sigquit);
```

```
/* get child process */
pid = fork ();
if (pid < 0) {
    perror("fork");
    exit(1); }
if (pid == 0) { for (;;);  }
else {
  printf("\nPARENT: sending SIGHUP\n\n");
  kill(pid, SIGHUP);
  sleep(3); // pause for 3 secs
  printf("\nPARENT: sending SIGINT\n\n");
  kill(pid, SIGINT);
  sleep(3); // pause for 3 secs
  printf("\nPARENT: sending SIGQUIT\n\n");
  kill(pid, SIGQUIT);
  sleep(3); }}
```

Παράδειγμα 5. Ανταλλαγή σήματος μεταξύ γονικής και θυγατρικής διεργασίας – Παράδειγμα Β (αρχείο sigExample5.c).

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <unistd.h>

// SIGALARM = 14
// SIGCHLD =  17

void signalHandler(int signal) {
   printf("Cought signal %d!\n",signal);
   if (signal==SIGCHLD) {
          printf("Child ended\n");
          wait(NULL); }}

int main() {
         signal(SIGALRM,signalHandler);
         signal(SIGUSR1,signalHandler);
         signal(SIGCHLD,signalHandler);
         if (!fork()) {
            printf("Child running...\n");
            sleep(2);
            printf("Child sending SIGALRM...\n");
            kill(getppid(),SIGALRM);
            sleep(5);
            printf("Child exitting...\n");
            return 0; }
         printf("Parent running, PID=%d. Press ENTER to exit.\n",getpid());
         getchar();
         printf("Parent exitting...\n");
         return 0; }
```

Παράδειγμα 6. Ανταλλαγή σήματος μεταξύ γονικής και πολλών θυγατρικών διεργασιών (αρχείο sigExample6.c).

```
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>

#define NUMPROCS 4    /* number of processes to fork */
int nprocs;           /* number of child processes */

void child (int n) {
    printf("\tChild[%d]: child pid=%d, sleeping for %d seconds\n", n, getpid(), n);
    sleep(n);
    printf("\tchild[%d]: I'm exiting\n", n);
    exit(100+n); }

void catch (int snum) {
    int pid, status;
    pid = wait(&status);
    printf("Parent process: child process pid=%d exited with value %d\n",
              pid, WEXITSTATUS(status));
    nprocs--;
    signal(SIGCHLD, catch); }

int main (int argc, char **argv) {
    int pid, i;
    signal(SIGCHLD, catch);
    for (i=0;i<NUMPROCS;i++) {
         pid=fork();
         if (pid<0) { perror("fork"); exit(1); }
         if (pid==0) child(i);
         else nprocs++; }
    printf("Parent process: going to sleep\n");
    while (nprocs != 0) {
            printf("parent: sleeping\n");
            sleep(60); }
    printf("Parent process: exiting\n");
   exit(0); }
```

Παράδειγμα 7. Παράδειγμα χρήσης των συναρτήσεων διαχείρισης σήματος – Παράδειγμα Α (αρχείο sigExample7.c).

```
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
static int got_signal = 0;
```

```
static void hdl (int sig) {
    got_signal = 1; }

int main (int argc, char *argv[]) {
    sigset_t mask;
    sigset_t orig_mask;
    struct sigaction act;
    memset (&act, 0, sizeof(act));
    act.sa_handler = hdl;
    if (sigaction(SIGTERM, &act, 0)) {
       perror ("sigaction");
      return 1; }
    printf ("Emptying signal set...\n");
    sigemptyset (&mask);
    printf ("Adding SIGTERM to signal set...\n");
    sigaddset (&mask, SIGTERM);
    // SIGTERM signal is added to orig_mask
   printf ("Adding signal set to the original mask...\n");
   if  (sigprocmask(SIG_BLOCK, &mask, &orig_mask) < 0) {
        perror ("sigprocmask");
        return 1; }
   // SIGTERM signal is blocked
   printf ("Blocking SIGTERM signal...\n");
   if  (sigprocmask(SIG_SETMASK, &orig_mask, NULL) < 0) {
       perror ("sigprocmask");
       return 1; }
   printf ("Sleeping for 2 seconds\n");
   sleep (2);
   if (got_signal) puts ("Got signal");
   return 0; }
```

Παράδειγμα 8. Παράδειγμα χρήσης των συναρτήσεων διαχείρισης σήματος – Παράδειγμα Β (αρχείο sigExample8.c).

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void handle_signal(int signal);
void handle_sigalrm(int signal);
void do_sleep(int seconds);

int main() {
   struct sigaction sa;
   printf("My pid is: %d\n", getpid()); // we need the pid to use kill from another terminal
   sa.sa_handler = &handle_signal; // here we assign the signal handler
   sa.sa_flags = SA_RESTART;
   // Block every signal during the handler
   sigfillset(&sa.sa_mask);
   // Signals SIGHUP, SIGUSR1 and SIGINT are associated with struct sa
```

```
   if (sigaction(SIGHUP, &sa, NULL) == -1) { perror("Error: cannot handle SIGHUP"); }
   if (sigaction(SIGUSR1, &sa, NULL) == -1) { perror("Error: cannot handle SIGUSR1"); }
   if (sigaction(SIGINT, &sa, NULL) == -1) { perror("Error: cannot handle SIGINT"); }
   // Will always fail, SIGKILL is intended to force kill your process
   if (sigaction(SIGKILL, &sa, NULL) == -1) { perror("Cannot handle SIGKILL"); }
   for (;;) {
      printf("\nSleeping for ~3 seconds\n");
      do_sleep(3); }}

void handle_signal (int signal) {
   const char *signal_name;
   sigset_t pending;
   // Find out which signal we're handling
   switch (signal) {
      case SIGHUP:
         signal_name = "SIGHUP";
         break;
      case SIGUSR1:
         signal_name = "SIGUSR1";
         break;
      case SIGINT:
         printf("Caught SIGINT, exiting now\n");
         exit(0);
      default:
         fprintf(stderr, "Caught wrong signal: %d\n", signal);
         return; }
   // However, printf in signal handlers IS NOT reccommented !
   printf("Caught %s, sleeping for ~3 seconds\n"
        "Try sending another SIGHUP / SIGINT / SIGALRM "
        "(or more) meanwhile\n", signal_name);
   do_sleep(3);
   printf("Done sleeping for %s\n", signal_name);
   // So what did you send me while I was asleep?
   sigpending(&pending);
   if (sigismember(&pending, SIGHUP)) { printf("A SIGHUP is waiting\n"); }
   if (sigismember(&pending, SIGUSR1)) { printf("A SIGUSR1 is waiting\n"); }
   printf("Done handling %s\n\n", signal_name); }

void handle_sigalrm(int signal) {
   if (signal != SIGALRM) { fprintf(stderr, "Caught wrong signal: %d\n", signal); }
   printf("Got sigalrm, do_sleep() will end\n"); }

void do_sleep(int seconds) {
   struct sigaction sa;
   sigset_t mask;
   sa.sa_handler = &handle_sigalrm; // Intercept and ignore SIGALRM
   sa.sa_flags = SA_RESETHAND; // Remove the handler after first signal
   sigfillset(&sa.sa_mask);
   sigaction(SIGALRM, &sa, NULL);
   // Get the current signal mask
   sigprocmask(0, NULL, &mask);
```

```
// Unblock SIGALRM
sigdelset(&mask, SIGALRM);
// Wait with this mask
alarm(seconds);
sigsuspend(&mask);
printf("sigsuspend() returned\n"); }
```