Sistemas Operativos
2021 / 2022

Licenciatura em Engenharia Informática

Lab. 07 – Comunicação assíncrona entre processos – sinais

Nesta aula pretende-se que os alunos fiquem com uma noção prática da comunicação assíncrona, utilizando *signals* no sistema operativo Linux.

Ex. 1 – Utilização de *signals*

Compile e execute o seguinte programa, e verifique o seu funcionamento.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <sys/wait.h>
5  #include <signal.h>
6
7  void signal_handler(int signal)
8  {
9      printf("Got signal %d..\n", signal);
10 }
11
12 int main()
13 {
14     signal(SIGUSR1, signal_handler);
15
16     int pid = fork();
17     if (pid == 0) {
18         printf("Hello from child!\n");
19         sleep(2);
20         exit(0);
21     } else {
22         printf("Hello from parent!\n");
23         kill(pid, SIGUSR1);
24         wait(NULL);
25     }
26     return 0;
27 }
```

Ex. 2 – Espera activa

Edite e compile um programa que crie um processo filho e espere por sinais vindo do processo pai. O processo filho deverá bloquear numa espera activa usando o ciclo *while(1)*. Por sua vez o processo pai deverá enviar 4 sinais ao filho com 1 segundo de intervalo entre cada sinal: SIGUSR1, SIGUSR2, SIGINT e por fim SIGKILL.

Ex. 3 – Signal handler

Edite e compile o seguinte programa:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <sys/wait.h>
5  #include <signal.h>
6
7  int i, pid;
8
9  void signal_handler(int signal)
10 {
11     printf("Parent: Handling process #%d with PID=%d\n", i, pid);
12 }
13
14 int main()
15 {
16     signal(SIGUSR1, signal_handler);
17
18     for (i=0; i<5; i++) {
19         int pid = fork();
20         if (pid == 0) {
21             printf("Child #%d, PID=%d\n", i, getpid());
22             kill(getppid(), SIGUSR1);
23             sleep(2);
24             printf("Child #%d, PID=%d is exiting\n", i, getpid());
25             exit(0);
26         }
27         sleep(1);
28     }
29
30     sleep(5);
31
32     for (int i=0; i<5; i++) {
33         int pid = wait(NULL);
34         printf("Parent: child with PID=%d ended\n", pid);
35     }
36
37     return 0;
38 }
```

a) Explique o seu funcionamento.

b) Altere o programa de modo que o processo pai envie para cada filho um sinal SIGUSR2. Cada filho deve receber esse sinal, escrever o seu valor de ordem de processo (valor da variável **i**) e o seu PID, e finalmente terminar a sua execução dentro da função que trata o sinal SIGUSR2.

Sugestão: Considere o seguinte exemplo em que os *pids* dos processos filhos estão a ser guardados num array à medida que está sendo feito o *fork*.

```
int main()
{
    int pids[5];

    for (i=0; i<5; i++) {
        pids[i] = fork();
        if (pids[i] == 0) {
            ...
        }
    }

    ...

    return 0;
}
```

(fim de enunciado)