PL2 - Signals

Luís Nogueira, Orlando Sousa

{lmn, oms}@isep.ipp.pt

Sistemas de Computadores
2024/2025

1. Consider the following code:

```
1  void handle_signal(int sig){
2     execlp("prog", "prog", NULL);
3  }
4
5  int main(){
6     pid_t pid;
7     int i;
8     struct sigaction act;
9
10    memset(&act, 0, sizeof(struct sigaction));
11    act.sa_handler = handle_signal;
12    act.sa_flags = SA_RESTART;
13    sigaction(SIGUSR1, &act, NULL);
14
15    pid = fork();
16    if(pid == 0){
17       for(i=0; i<5; i++){
18          kill(getppid(), SIGUSR1);
19          sleep(2);
20       }
21    }else{
22       for(; ;)
23          pause();
24    }
25 }
```

a) How many times is the program "prog" executed? Provide a justification for your answer.

2. Write a program that accepts two integers as arguments and divides the first number by the second. When the divisor is zero, the program should catch the corresponding signal instead of crashing.

3. Consider a program that reads memory location 0, which is an invalid location for user processes:

```
1 int main() {
2   int a;
3   a = *(int *)0;
4 }
```

   The program will cause a segmentation fault. Your task is to add a signal handler to trap the signal. The desired output should be: "I got signal SIGSEGV".

4. Implement a program to populate an array of 100000 positions without repeating values. Then, spawn 5 new child processes. The objective is to concurrently determine whether a given number is present in the array.

   - Each child process handles 20000 positions.
   - The process that discovers the number should print the position where it was found and exit with its child number (1, 2, 3, 4, or 5).
   - Processes that do not find the number should exit with the value 0;
   - The parent process must wait for all children to terminate and print the PID and number of the child where the number was found, or an error message if the value was not found in the array.

   Provide an efficient solution to the problem that terminates the processes still executing after the number has been found.

5. Develop the `schedule` program that creates and schedules a set of processes. The programs executed by the child processes are indicated in the command line as follows:

   `schedule prog1 prog2 ... progn`

   Within the `schedule` program, only one child process is allowed to run at any given time. Each child process runs for 5 consecutive seconds before being suspended, allowing the next process to execute in a cyclic manner. This behavior should be achieved using `SIGSTOP` and `SIGCONT` signals.

6. Consider an array in which each position contains the name of a command and the maximum execution time. The array consists of structures of the following type:

```
typedef struct{
  char cmd[32];
  int time;
}command_t;
```

Implement a program that:

- Sequentially executes all commands stored in the array.
- If a command exhausts its maximum execution time, it should be terminated.

Use process primitives, signals, and exec functions.

7. A software company intends to test the viability of applying a new algorithm in data processing to discover certain types of information. To do this, it wants to use this new algorithm in a simulation that employs parallel/concurrent programming. The application to be developed should create 50 processes and test the functionality of the new algorithm as follows:

   - Each child process begins by executing the function `simulate1()`. This function returns 1 if it finds the data and returns 0 if it does not.
   - After completing the execution of the function, each child process should send, to the parent process, the `SIGUSR1` signal, if the function was successful, or the `SIGUSR2` signal if not.
   - Then, it should wait for the parent to send it a signal to start the function `simulate2()`.
   - The parent process should manage the simulation, performing the following test after 25 processes have finished `simulate1()`:
     - If until that moment none of the searches have been successful, the parent process should print "Inefficient Algorithm!" and terminate all children.
     - If until that moment there is at least one process that has performed a successful search, the parent process should send a signal to all its children, indicating that they should start executing the function `simulate2()`, eventually interrupting the execution of the function `simulate1()`.

   Note: For the implementation of the `simulate1()` and `simulate2()` functions, use random numbers to simulate the needed computation time and the end result of that computation.

8. To enhance the response time for clients of a car assistance company, you've been assigned the task of developing an application that uses 10 processes to search the database for the workshop in the region where a specific client is located.

   - The application prompts for the name of the region where the client is situated;
   - Upon locating the region, a process should print the name of the workshop and its corresponding phone number;
   - If the region has already been found, the parent process terminates the execution of those children processes that have not yet completed.

   Consider that the database contains 10000 entries, and each entry structured as follows:

```
typedef struct {
  char name[50];
  char region[50];
  int phone;
}record_t;
```

9. Implement a program that, after spawning a new process, follows this cyclic pattern:
   - The parent process sleeps for 2 seconds and then sends a SIGUSR1 signal to its child.
   - The child process, upon receiving the signal, prints on the screen: "Handled SIGUSR1". The handler associated with `SIGUSR1` should not be interrupted by the reception of a `SIGUSR2`.

   Terminate both processes in the command line using the `kill` command.

10. Implement a program that prompts the user to enter a sentence. If no input is received within 10 seconds, the program should print the message: "You are too slow!".

    a) Implement a solution that addresses the problem using a single process.

    b) Implement a solution that employs two processes to handle the time limit.

    Ensure your program ignores signals `SIGUSR1` and `SIGUSR2`.

11. Implement a program that simulates the `sleep(n)` function. Ensure that your program blocks the reception of all signals except for the one selected to solve the problem.

12. Write a program that, upon receiving the `SIGUSR1` signal, prints the message "I captured a SIGUSR1 sent by the process with PID XX", where XX is the PID of the process which sent the `SIGUSR1` signal.

    Suggestion: You can use `snprintf()` to format the text into a character array, and then use the `write()` system call to output that text, as shown in the example below:

```
1 void foo(){
2   char msg[80];
3   int length;
4
5   length = snprintf(msg, 80, "Here is my PID: %d\n", getpid());
6   write(STDOUT_FILENO, msg, length+1);
7 }
```

13. Access to a shared resource demands synchronisation. Implement a program that spawns a new process and synchronises access to a "data.txt" text file using signals.

- The parent process waits for a `SIGUSR1` signal to arrive; upon receiving the signal, it should read a number from the file;
- After reading that number, the parent displays it on the screen;
- Finally, it sends a `SIGUSR2` signal to the child process;
- The child process is responsible for clearing the content of the file, writing a new number to the file, and signaling the parent.

Notes: The writing and reading operations should be performed cyclically. Both processes should block the reception off all signals, except `SIGUSR1` and `SIGUSR2`. If `SIGINT` is pending, they should terminate.