Signals

Samuele Germiniani
samuele.germiniani@univr.it

UNIVERSITÀ di VERONA
Department of ENGINEERING FOR INNOVATION MEDICINE

Cyber-Physical & IoT Systems Design

Introduction - Part 1

A signal is a notification to a process that an event has occurred. They interrupt the normal flow of execution of a program.

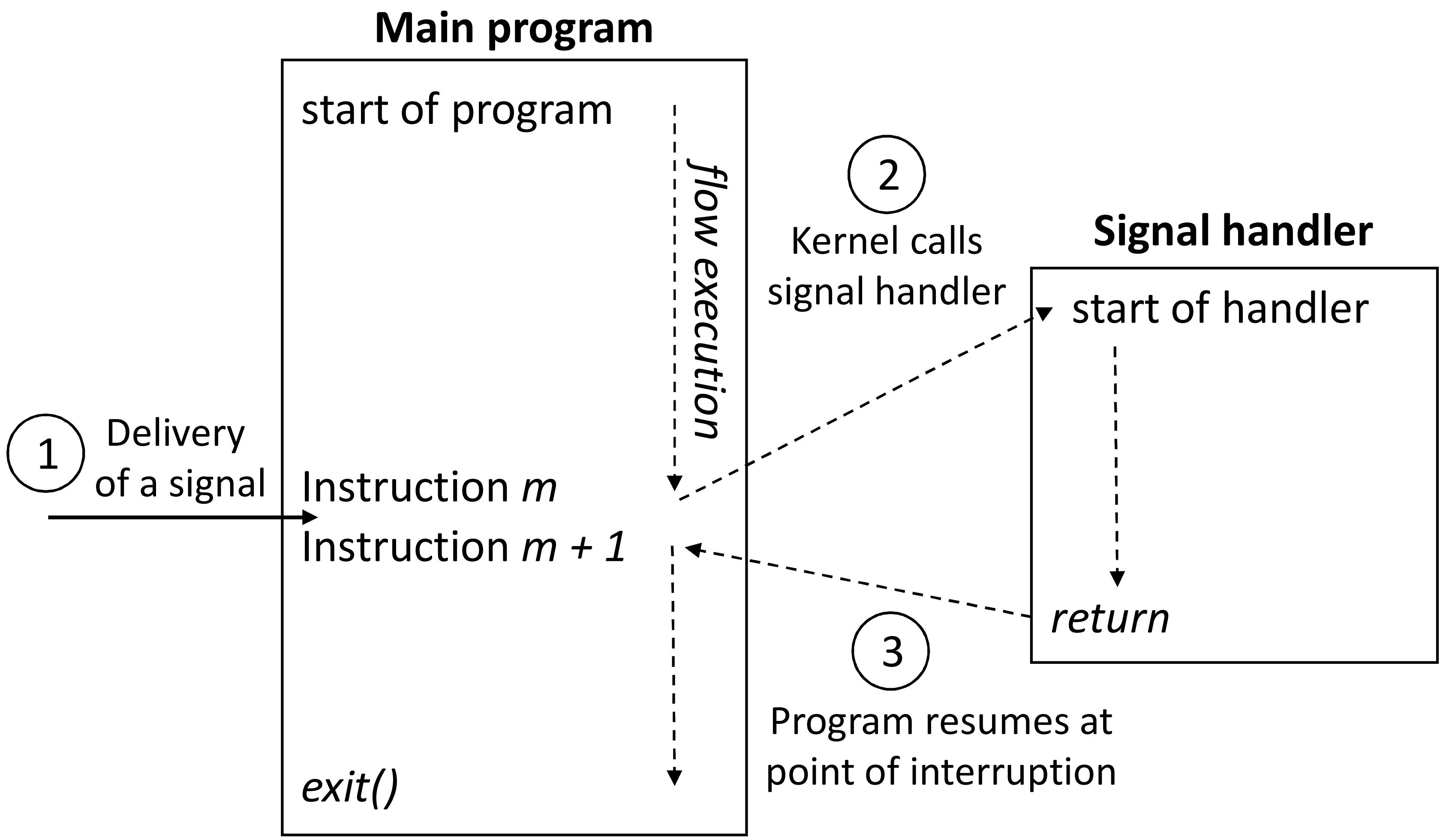
- A signal is generated by an event
- After being generated, a signal is later delivered to a process
- Between the time it is generated and the time it is delivered, a signal is said to be pending.
- Normally, a pending signal is delivered to a process as soon as it is next scheduled to run, or immediately if the process is already running.

Introduction - Part 2

Upon delivery of a signal, a process carries out one of the following default actions, depending on the signal:

- The process is terminated (killed).
- The process is suspended (stopped).
- The process is resumed after previously being stopped.
- The signal is ignored. It is discarded by the kernel and has no effect on the process.
- The process executes a signal handler, namely a function written by the programmer that performs appropriate tasks in response to the delivery of a signal.

Signal handler



Invocation of a signal handler may interrupt the main program flow at any time.

- The kernel calls the signal handler, and when the handler returns, execution of the program resumes at the point where the handler interrupted it.

Signal handler - example

```
void sigHandler(int sig) {
    printf("The signal %s was caught!\n",
        (sig == SIGINT)? "Ctrl-C" : "signal User-1");
}
```

Exercise templates

- Download the templates of the exercises

```
git clone https://github.com/SamueleGerminiani/ex_signals_templates
```

Exercise 1

Specifications

- Write a program that continuously prints to the screen every 5 seconds the string "Hello, I am a running process (PID)" (PID is the process identifier of the running process).
- The program should block all system signals except the SIGINT signal.
- Whenever SIGINT is sent to the running process, it should print the string "Ctrl+C cannot stop me!" to the screen.
- N.B. To terminate the running process, open a new terminal and send the SIGKILL signal to the process using the command kill -SIGKILL PID (PID is the process identifier of the running process).

Hints (not ordered)

```
sigfillset(…);
sigdelset(… , …);
sigprocmask(… , … , …);
SIG_ERR,  SIGINT
signal(… , …)
```

Exercise 2

Specifications

- Write a program that implements the "sleep" command with the specified number of seconds (see man sleep) using the system calls alarm(...) and pause().
- When the specified number of seconds has elapsed, the program prints the string "What a nice nap!" to the screen.

Hints

signal(… , …)
SIGALRM, SIG_ERR
alarm(…)
pause()

Exercise 3

Specifications

- Write a program that generates two child processes.
- Child process 1 sends the SIGUSR1 signal to the parent process.
- The parent process forwards the SIGUSR1 signal to child process 2.
- After receiving SIGUSR1, child process 2 prints "<child2> SIGUSR1 received!" to the screen, sends the SIGUSR2 signal to the parent process, and finally terminates.
- The parent process forwards the SIGUSR2 signal to child process 1.
- After receiving SIGUSR2, child process 1 prints "<child1> SIGUSR2 received!" to the screen and finally terminates.
- The parent process waits for the termination of child 1 and child 2 and finally terminates.

Hints (not ordered)

sigfillset(…)
signal(… , …)
sigprocmask(… , … , …);
SIG_ERR, SIGUSR2, SIGUSR1
sigdelset(…,…);
getppid()
kill(…,…)
wait(…)

Exercise 4

Specifications

- Write a program that generates two child processes.
- Child process 1 continuously prints the string "I am child 1, I am playing..." to the screen every second.
- Child process 2 sends the SIGSTOP signal to child 1 every 3 seconds continuously (disturbance action).
- The parent process continuously monitors the created child processes every 5 seconds (monitoring action).
- If child 1 is in the STOP state, the parent process sends the SIGCONT signal to child 1 (restoration action).
- If the user sends the SIGINT signal to the parent process, the parent process terminates the two child processes by sending the SIGTERM signal and finally terminates itself.

Hints (not ordered)

sleep(…)
printf(…)
kill(…,…)
signal(…,…)
SIGSTOP, SIG_INT, SIG_ERR
waitpid(…,…,…)
WIFSTOPPED(…)