SYS2(OS) Week 5 Lab4b: Process Signals

Dr Poonam Yadav and Sven Signer

Process States

Orphan Process

- Occurs when the parent process terminates before the child.
- The OS assigns the child a new parent process further up the process tree.
- Historically this was typically the init process with PID 1, but on modern systems it may be some other intermediate process.

Zombie Process

- Occurs when a child process has terminated, but the parent has not yet waited for it.
- Zombie process remains in the OS process table to preserve information that the parent might request later.
- After it has been waited on, the OS can reap the zombie process.

Signals

Signals allow processes to be notified of certain conditions and can be triggered in different was.

- Automatically by the operating system.
- Programmatically by another process.
- Manually by the user

Some signals might already be familiar, though you might not yet have realised they are signals.

- Keyboard interrupt (SIGINT): Triggered if the user presses Ctrl-C in a terminal.
- Segmentation fault (SIGSEGV): Sent by the operating system if the process attempts an illegal memory access.

Different signals have different meanings. As usual, they are explained in the man pages: `man 7 signal`

Signal System Call

Description

- Allows the programmer to register custom signal handlers for most signals.
- Some signals, such as keyboard interrupt (SIGINT) make sense to have custom behaviour. Others, such as segmentation fault usually less so.
- The user signals (SIGUSR1/SIGUSR2) are reserved for the application developer to assign their own meaning.

Call Signature

```
#include <signal.h>
signal(int sig,
       void (*func)(int));
```

- Takes two arguments: a signal type ID and a function pointer to the signal handler function. A function pointer can simply be the name of a function with the correct argument/return types.
- See manual page for further details: `man 2 signal`

Signal Example

```c
#include <stdio.h>
#include <signal.h>

void interrupt_handler(int signal) {
    printf("Handling graceful exit\n");
    ...
    exit(0);
}


int main() {
    signal(SIGINT, interrupt_handler);

    ...
}
```

Define an example handler function that could gracefully shutdown the application when a keyboard interrupt signal is received.

Register the function as the signal handler for SIGINT at the start of program execution.

Kill System Call / Utility

Description

- Allows a program to send a signal to another program.
- For example: parent process might terminate the child process if it is no longer needed.

Call Signature

-
  ```
  #include <signal.h>
  int kill(pid_t pid, int sig);
  ```
- Sends signal sig to process with the specified process ID.
- Full description: `man 2 kill`

Command Line Utility

- Like many system calls, it has an equivalent command line utility to manually trigger it.
- `$ kill -signal PID`
- Full description: `man 1 kill`

Assignment

This lab session looks at understanding the behaviour of new processes created with the fork system call.

An example program `simple_fork.c` is provided on the VLE to get you started.