Linux Processes

CSCI 3753: Operating Systems
Fall 2017

1. Process Creation

A new process can be created from inside a C/C++ program by using the *fork()* system call. The process that calls *fork()* is called the parent process, and the newly created process is called the child process. After the call, the parent process and the child process run concurrently.

```
#include <sys/types.h>
#include <unistd.h>

pid_t pid;
pid = fork();
```

The child process is an exact copy of the parent process, except for the following:

- The child process has a unique process id (PID).
- The child process has a different process id of its creator (PPID).

The fork is called by the parent process, but returns in both the parent and the child processes. In the parent process, the return value is the process id of the child process, whereas in the child process, the return value is 0. If fork fails, no child process is created and -1 is returned in the process calling fork.

```
pid t pid;
if ((pid = fork()) == -1) exit(1); /* FORK FAILED */
if (pid == 0) {
/*put code for child process here*/
   cout << "I am the child process." << endl;
   cout << "My process id = " << getpid() << endl;
   cout << "My parent's process id = " << getppid() << endl;
   exit(0);
}
/*put code for parent process here*/
cout << "I am the parent process." << endl;
cout << "My process id = " << getpid() << endl;
```

2. Process Execution Image

The child process is not restricted to executing a subset of the statements in the parent process. It can also execute another program by overlaying itself with an executable file. The target executable file is read in on top of the address space of the very process that is executing, overwriting it in memory, and execution continues at the entry point defined in the file. The result is that the child process begins to execute a new program, under the same execution environment as the old program, which is now replaced.

The program overlay can be done by any of the several versions of exec system calls, including execl, execv, and execve. These calls differ from one another in the type and number of arguments they take. Read the man page of exec for details.

```
pid t pid;
if ((pid = fork()) == -1) exit(1); /* FORK FAILED */
if (pid == 0) {
/* Child process will execute the executable code in the file a.out */
   execl("a.out", NULL);
   exit(0);
}
/*put code for parent process here*/
```

3. Synchronization between Parent and Child Processes

After creating a child process, the parent and the child processes run independentl of one another. At any point during execution, the parent process can elect to wait for the child process to terminate, by using the wait system call, before proceeding further.

```
#include <sys/types.h>
#include <sys/wait.h>
int pid = wait(union wait *status)
```

The wait system call searches for a terminated child of the calling process.

1. If there are no child processes, wait returns immediately with value -1.
2. If one or more child processes have terminated already, wait selects an arbitrary terminated child, stores its exit status in the variable status, and returns its process id.
3. Otherwise, wait blocks until one of the child processes terminates and then goes to step 2.

4. Process Termination

Every running process eventually terminates.

1. The process runs to completion and the function main returns.
2. The process calls the library routine exit, or the system call exit.
3. The process encounters an execution error or receives an interrupt signal, causing its premature termination.

The argument to _exit/exit is part of the termination status of the process. Conventionally, a zero argument indicates normal termination and a non-zero argument indicates abnormal termination.

5. Process Suspension

Execution of a process may be suspended for some interval of time using the sleep command.

#include <unistd.h> unsigned seconds;
unsigned r = sleep(seconds);

6. Some Useful Commands

ps: The ps command displays information about the existing processes in the system. This command has several command-line arguments. Read the man page for details.

kill pid: This command terminates an existing process whose process id is pid.