COMP1521 Tutorial 08

Process Forking

What do these syscalls do? What are the return values and when they might occur

1. pid_t fork(void);
2. int execve(char *filename, char *argv[], char *envp[]);

Linux Commands

You can pass files into programs in UNIX in two ways

- $ cat file
- $ cat < file

Describe how each of these cases might be implemented in code.

Process Forking

What are the possible outputs from this code.

You can assume that all of the appropriate #includes have been done.

```
1. int main(void)
2. {
3.     int x = 1;
4.     pid_t pid = fork();
5.     if (pid < 0)
6.        { fprintf(stderr, "Fork failed\n"); exit(1); }
7.     elseif (pid > 0)
8.        { x++;  printf("x = %d\n", x); }
9.     else
10.       { x--;  printf("x = %d\n", x); }
11. }
```

Program Execution

Each new process in a computer system will have a new address space.

Which parts of the address space contain initial values at the point when the process starts running? Code? Data? Heap? Stack?

Which parts of the address space can be modified as the process executes?

Processes

- What does each of the columns represent?
- What do the first characters in the STAT column mean?
- Which process has consumed the most CPU time?

```
  PID    VSZ   RSS TTY      STAT START   TIME COMMAND
    1   3316  1848 ?        Ss   Jul08   1:36 init
  321   6580  3256 pts/52   Ss+  Aug26   0:00 -bash
  334  41668 11384 pts/44   Sl+  Aug02   0:00 vim timing_result.txt
  835   6584  3252 pts/124  Ss+  Aug27   0:00 -bash
  857  41120 10740 pts/7    Sl+  Aug22   0:00 vi echon.pl
  924   6524  3188 pts/184  Ss   15:52   0:00 -bash
  938   3664    96 pts/184  S    15:52   0:00 /usr/local/bin/checkmail
 1199   6400  3004 pts/142  Ss   Oct05   0:00 -bash
 1381  41504 11436 pts/142  Sl+  Oct05   0:00 vim PageTable.h
 2558   3664    96 pts/120  S    13:47   0:00 /usr/local/bin/checkmail
 2912  41512 11260 pts/46   Sl+  Aug02   0:00 vim IntList.c
 3483  14880  5168 pts/149  S+   Sep20   0:00 gnuplot Window.plot
 3693  41208 11240 pts/120  Tl   13:50   0:00 vim trace4
 3742   6580  3320 pts/116  Ss+  Sep07   0:00 -bash
 5531   6092  2068 pts/158  R+   16:04   0:00 ps au
 5532   4624   684 pts/158  S+   16:04   0:00 cut -c10-15,26-
 5538   3664    92 pts/137  S    15:05   0:00 /usr/local/bin/checkmail
 6620   5696  3028 pts/89   S+   Aug13   0:00 nano PingClient.java
 7132  41516 11196 pts/132  Sl+  Sep08   0:00 vim board1.s
12256 335316 10436 ?        Sl   Aug14  15:01 java PingServer 3331
12272   4260  2816 ?        Ss   Aug02  10:34 tmux
12323  10276  4564 ?        S    Sep09   0:02 /usr/lib/i386-linux-gnu/gconf/gconfd-2
12461   4260  2808 ?        Ss   Sep02   5:42 tmux
13051  43448 13320 pts/110  Sl+  Sep05   0:02 vim frequency.pl
13200  47772 21928 ?        Ssl  15:19   0:02 gvim browser.cgi
13203  41756 11560 pts/26   Sl+  Aug12   0:02 vim DLList.h
13936  11872  6856 ?        S    Sep19   0:06 /usr/lib/gvfs/gvfs-gdu-volume-monitor
30383   7624  3828 pts/77   S+   Aug23 336:28 top
```

Processes

- Why do some processes have no TTY?
- When was this machine last re-booted?

```
  PID    VSZ   RSS TTY      STAT START   TIME COMMAND
    1   3316  1848 ?        Ss   Jul08   1:36 init
  321   6580  3256 pts/52   Ss+  Aug26   0:00 -bash
  334  41668 11384 pts/44   Sl+  Aug02   0:00 vim timing_result.txt
  835   6584  3252 pts/124  Ss+  Aug27   0:00 -bash
  857  41120 10740 pts/7    Sl+  Aug22   0:00 vi echon.pl
  924   6524  3188 pts/184  Ss   15:52   0:00 -bash
  938   3664    96 pts/184  S    15:52   0:00 /usr/local/bin/checkmail
 1199   6400  3004 pts/142  Ss   Oct05   0:00 -bash
 1381  41504 11436 pts/142  Sl+  Oct05   0:00 vim PageTable.h
 2558   3664    96 pts/120  S    13:47   0:00 /usr/local/bin/checkmail
 2912  41512 11260 pts/46   Sl+  Aug02   0:00 vim IntList.c
 3483  14880  5168 pts/149  S+   Sep20   0:00 gnuplot Window.plot
 3693  41208 11240 pts/120  Tl   13:50   0:00 vim trace4
 3742   6580  3320 pts/116  Ss+  Sep07   0:00 -bash
 5531   6092  2068 pts/158  R+   16:04   0:00 ps au
 5532   4624   684 pts/158  S+   16:04   0:00 cut -c10-15,26-
 5538   3664    92 pts/137  S    15:05   0:00 /usr/local/bin/checkmail
 6620   5696  3028 pts/89   S+   Aug13   0:00 nano PingClient.java
 7132  41516 11196 pts/132  Sl+  Sep08   0:00 vim board1.s
12256 335316 10436 ?        Sl   Aug14  15:01 java PingServer 3331
12272   4260  2816 ?        Ss   Aug02  10:34 tmux
12323  10276  4564 ?        S    Sep09   0:02 /usr/lib/i386-linux-gnu/gconf/gconfd-2
12461   4260  2808 ?        Ss   Sep02   5:42 tmux
13051  43448 13320 pts/110  Sl+  Sep05   0:02 vim frequency.pl
13200  47772 21928 ?        Ssl  15:19   0:02 gvim browser.cgi
13203  41756 11560 pts/26   Sl+  Aug12   0:02 vim DLList.h
13936  11872  6856 ?        S    Sep19   0:06 /usr/lib/gvfs/gvfs-gdu-volume-monitor
30383   7624  3828 pts/77   S+   Aug23 336:28 top
```

```
print a prompt
while (read another command line) {
   break the command line into an array of words (args[])
   // args[0] is the name of the command, a[1],... are the command-line args
   if (args[0] starts with '.' or '/')
      check whether args[0] is executable
   else
      search the command PATH for an executable file called args[0]
   if (no executable called args[0])
      print "Command not found"
   else
      execute the command
   print a prompt
}
```

UNIX/Linux shells

- Text prompt that allows you to run other programs
- How can you find what directories are in the PATH?
- Describe the "search the command PATH" process in more detail, including the kinds of system calls that would be needed to determine whether there was an executable file in one of the path directories?

More Forking

What are the possible outcomes of this code?

When might fork() fail?

```c
int main(void)
{
    printf("Hello\n");
    if (fork() != 0)
        printf("Gan bei\n");
    else
        printf("Prost\n");
    printf("Goodbye\n");
}
```

Execve()

The function execve() aims to replace the current process with a process executing the specified program. Ideally, the execve() function never returns. However, it has a return type of int.

1. Under what circumstances would does execve() return a value?
2. What should the calling program do if execve() does return?
3. Write two small pograms to work out whether execve() causes a new process ID to be created, or whether it inherits the caller's process ID.

Execve()

- Used for running binaries
- Replaces current running process with the new one
- Does not return if successful, returns -1 if unsuccessful
- The process still maintains the same process ID

Signals and Kill()

Kill command and kill() system call can be used to send a signal to a specified process, what do these signals do:

1. SIGHUP
2. SIGINT
3. SIGABRT
4. SIGFPE
5. SIGSEGV
6. SIGPIPE
7. SIGTSTP
8. SIGCONT

Signal Handling

Sigaction() used for handling signals takes 3 arguments (make the program do something else when receiving signals)

- Int signum – signal which this is defined for handling
- Struct sigaction *act – pointer to record describing how to handle the signal
- Struct sigaction *oldact – pointer to record describing how the signal was handled

Sigaction

What does this program do if it receives

- SIGHUP signal
- SIGINT signal
- SIGTSTP signal
- SIGKILL signal

```
1. // assume a bunch of #include's
2.
3. void handler(int sig)
4. {
5.     printf("Quitting...\n");
6.     exit(0);
7. }
8.
9. int main(int argc, char *argv[])
10. {
11.     struct sigaction act;
12.     memset (&act, 0, sizeof(act));
13.     act.sa_handler = &handler;
14.     sigaction(SIGHUP, &act, NULL);
15.     sigaction(SIGINT, &act, NULL);
16.     sigaction(SIGKILL, &act, NULL);
17.     while (1) sleep(5);
18.     return 0
19. }
```