CS431 — Project 1

April 6, 2017

Due: Friday, April 14, 2017 by midnight (50 points)

Description

In this project you will simulate a process table and some system calls. I will assume you are implementing this in Java but you can use a different language. If you use a different language, I need complete compilation and testing procedures or a working Makefile. If using Java, your program should compile and run with the following:

```
$ javac ProcessTable.java
$ java ProcessTable
```

You should implement a process table that is a list of process control blocks. Each process control block should have the following information:

1. An `int` referring to the process ID (pid).
2. A `String` referring to the program being executed.
3. A `String` referring to the user name running the program.
4. An `int` referring to the current status. A 0 indicates the program is running, a 1 indicates the process is ready, and a 2 indicates the process is blocked.
5. Six `int`s referring to register contents for the following registers:
   (a) `pc` (program counter)
   (b) `sp` (stack pointer)
   (c) `r0` (register 0)
   (d) `r1` (register 1)
   (e) `r2` (register 2)
   (f) `r3` (register 3)

You should also have a data structure recording the current CPU register contents (the same six registers stored for each process control block).

Your process table should start with a single process with pid 1, user `root`, program name `init`, and randomly generated register contents. This process should have status 0 (running) and you should put the same register contents in to your CPU data structure.

Your program should have a command line interface that allows the user to enter commands that will modify your process table and CPU. It should support the following commands:

1. `fork`: this command should make a copy of the currently running process with status 1 (ready) and a new unique pid. The program, user, and register contents should be the same as the running process.

2. `kill pid`: this command should kill the process with the specified process id only if either the currently running process has user `root` or it is the same user as the process being killed.
3. `execve program user`: this command should switch the program and user name for the currently running program to the values specified and set all the register contents to newly randomized values. This should modify both the CPU registers and process table entry for the process. Only `root` can call `execve` as `root` and `root` can call `execve` for any user.
4. `block`: this command should put the currently running process in to the blocked state (2). It should be unloaded from the CPU (put register contents in to the process table) and a new, randomly chosen ready process should be loaded to the CPU.
5. `yield`: this command should put the currently running process in to the ready state (1). It should be unloaded from the CPU (put the register contents in to the process table) and a new, randomly chosen ready process should be loaded to the CPU.
6. `exit`: this command should cause the currently running process to exit (remove it from the process table). A new, randomly chosen ready process should be loaded to the CPU.
7. `print`: this command should print out the CPU and process table contents as shown below in the sample output.
8. `unblock pid`: this command should move the blocked process with the specified pid to the ready state.

Here is some sample output for the program:

```
> print
CPU:
  PC = 0x595cdc6d    SP = 0x8c173e68
  R0 = 0x65be8283    R1 = 0x5bcb4c90
  R2 = 0x5a18cc9c    R3 = 0x7d8b2184

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      0 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184

> fork
> print
CPU:
  PC = 0x595cdc6d    SP = 0x8c173e68
  R0 = 0x65be8283    R1 = 0x5bcb4c90
  R2 = 0x5a18cc9c    R3 = 0x7d8b2184

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      0 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   2         init     root      1 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184

> yield
> print
CPU:
  PC = 0x595cdc6d    SP = 0x8c173e68
  R0 = 0x65be8283    R1 = 0x5bcb4c90
  R2 = 0x5a18cc9c    R3 = 0x7d8b2184

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      1 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
```

```
   2         init     root      0 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184

> execve javac someuser
> print
CPU:
  PC = 0x8e123446    SP = 0x86814f09
  R0 = 0xc72a3d48    R1 = 0x3a597af3
  R2 = 0x7be94f85    R3 = 0xb0d7d921

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      1 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   2        javac someuser      0 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921

> fork
> execve notepad root
> print
CPU:
  PC = 0x8e123446    SP = 0x86814f09
  R0 = 0xc72a3d48    R1 = 0x3a597af3
  R2 = 0x7be94f85    R3 = 0xb0d7d921

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      1 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   2        javac someuser      0 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921
   3        javac someuser      1 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921

> execve notepad someuser
> print
CPU:
  PC = 0x53054884    SP = 0x1b4f8fe7
  R0 = 0xc5383d58    R1 = 0x9552e7d3
  R2 = 0x607403a9    R3 = 0xdc62ca7f

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      1 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   2      notepad someuser      0 0x53054884 0x1b4f8fe7 0xc5383d58 0x9552e7d3 0x607403a9 0xdc62ca7f
   3        javac someuser      1 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921

> exit
> print
CPU:
  PC = 0x8e123446    SP = 0x86814f09
  R0 = 0xc72a3d48    R1 = 0x3a597af3
  R2 = 0x7be94f85    R3 = 0xb0d7d921

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      1 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   3        javac someuser      0 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921

> block
> print
CPU:
```

```
  PC = 0x595cdc6d    SP = 0x8c173e68
  R0 = 0x65be8283    R1 = 0x5bcb4c90
  R2 = 0x5a18cc9c    R3 = 0x7d8b2184

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      0 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   3        javac someuser      2 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921

> unblock 3
> print
CPU:
  PC = 0x595cdc6d    SP = 0x8c173e68
  R0 = 0x65be8283    R1 = 0x5bcb4c90
  R2 = 0x5a18cc9c    R3 = 0x7d8b2184

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      0 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   3        javac someuser      1 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921

> yield
> print
CPU:
  PC = 0x8e123446    SP = 0x86814f09
  R0 = 0xc72a3d48    R1 = 0x3a597af3
  R2 = 0x7be94f85    R3 = 0xb0d7d921

Process Table:
 PID      Program     User Status         PC         SP         R0         R1         R2         R3
   1         init     root      1 0x595cdc6d 0x8c173e68 0x65be8283 0x5bcb4c90 0x5a18cc9c 0x7d8b2184
   3        javac someuser      0 0x8e123446 0x86814f09 0xc72a3d48 0x3a597af3 0x7be94f85 0xb0d7d921

>
```

Submission

Submit the project to a repository on https://codebank.xyz called `CS431-P1`. You can commit and push as many times as you want prior to the deadline.