PennOS Demo Outline

Prof. Boon Thau Loo

Spring 2023

TA: ______________________________ Group Number: ______________________________

1 Kernel

Waiting, Zombies, and Orphans

- *Demo 1 (Zombie)*:

```
$ zombify &
[1] 2
$ ps
PID PPID PRI STAT CMD
  1    0  -1  B   shell
  2    1   0  R   zombify
  3    2   0  Z   zombie_child
  4    1   0  R   ps
$
```

- *Demo 2 (Zombie to Orphan)*:

```
$ zombify
^C
$
```

Below is the relevant part of the log:

```
[  20]  ZOMBIE      3   0     zombie_child
[  20]  SCHEDULE    2   0     zombify
[  20]  SIGNALED    2   0     zombify
[  20]  ZOMBIE      2   0     zombify
[  20]  ORPHAN      3   0     zombie_child
```

- *Demo 3 (Orphans)*: Run `orphanify` in the shell. Below is the relevant part of the log:

```
[  15]  ZOMBIE      2   0     orphanify
[  15]  ORPHAN      3   0     orphan_child
```

- *Demo 4 (Blocking Wait Stress Test)*:

```
$ hang
child_0 was spawned
child_1 was spawned
child_2 was spawned
child_3 was spawned
child_4 was spawned
child_5 was spawned
```

```
child_6 was spawned
child_7 was spawned
child_8 was spawned
child_9 was spawned
child_1 was reaped
child_2 was reaped
child_3 was reaped
child_4 was reaped
child_5 was reaped
child_6 was reaped
child_7 was reaped
child_8 was reaped
child_0 was reaped
child_9 was reaped
$ logout
```

The `hang` command spawns 10 children and block-waits for any of them in a loop until all of them are reaped. Note that the order in which the children get reaped is non-deterministic.

- *Demo 5 (Nonblocking Wait Stress Test)*:

```
$ nohang
child_0 was spawned
child_1 was spawned
child_2 was spawned
child_3 was spawned
child_4 was spawned
child_5 was spawned
child_6 was spawned
child_7 was spawned
child_8 was spawned
child_9 was spawned
child_0 was reaped
child_1 was reaped
child_2 was reaped
child_3 was reaped
child_4 was reaped
child_6 was reaped
child_7 was reaped
child_8 was reaped
child_5 was reaped
child_9 was reaped
$ logout
```

The `nohang` command spawns 10 children and nonblocking waits for any of them in a loop until all of them are reaped. Note that the order in which the children get reaped is non-deterministic.

- *Demo 6 (Recursive Spawn Stress Test)*:

```
$ recur
Gen_A was spawned
Gen_B was spawned
Gen_C was spawned
Gen_D was spawned
Gen_E was spawned
Gen_F was spawned
Gen_G was spawned
```

```
Gen_H was spawned
Gen_I was spawned
Gen_J was spawned
Gen_K was spawned
Gen_L was spawned
Gen_M was spawned
Gen_N was spawned
Gen_O was spawned
Gen_P was spawned
Gen_Q was spawned
Gen_R was spawned
Gen_S was spawned
Gen_T was spawned
Gen_U was spawned
Gen_V was spawned
Gen_W was spawned
Gen_X was spawned
Gen_Y was spawned
Gen_Z was spawned
Gen_Z was reaped
Gen_Y was reaped
Gen_X was reaped
Gen_W was reaped
Gen_V was reaped
Gen_U was reaped
Gen_T was reaped
Gen_S was reaped
Gen_R was reaped
Gen_Q was reaped
Gen_P was reaped
Gen_O was reaped
Gen_N was reaped
Gen_M was reaped
Gen_L was reaped
Gen_K was reaped
Gen_J was reaped
Gen_I was reaped
Gen_H was reaped
Gen_G was reaped
Gen_F was reaped
Gen_E was reaped
Gen_D was reaped
Gen_C was reaped
Gen_B was reaped
Gen_A was reaped
$ logout
```

The `recur` command recursively spawns itself 26 times and names the spawned processes `Gen_A` through `Gen_Z`. Each process is block-waited and reaped by its parent.

Scheduling

- *Demo 0*: Run `busy` and report what CPU% you should be getting from `top`. (100%)

- *Demo 1*: Run `busy&` (-1) and report what CPU% you should be getting from `top`. (50%)
- *Demo 2*: Run `busy&` (0) and report what CPU% you should be getting from `top`. (40%)
- *Demo 3*: Run `busy&` (1) and report what CPU% you should be getting from `top`. (∼31%)
- *Demo 4*: Run `sleep` and report what CPU% you should be getting from `top`. (0%)
- *Demo 5*: Run `busy&` (-1), `busy&` (0), `busy&` (0), `busy&` (1), `busy&` (1) and show in your logs that no starvation occurs as well as your priority ratios are being adhered to. (∼76%)

Continuity

Your scheduler should stay running during the Scheduling test cases; otherwise a flat deduction will be applied.

2 File System

- *Demo 1 (Format)*:

```
root@cis548Dev:~# pennfat
pennfat# mkfs minfs 1 0
pennfat# mkfs maxfs 32 4
pennfat#
root@cis548Dev:~# ls -l minfs maxfs
...
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~# hd maxfs
...
root@cis548Dev:~#
```

- *Demo 2 (One Block)*:

```
root@cis548Dev:~# head -c 256 /dev/urandom > demo2
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# cp -h demo2 f1
pennfat# ls
...
pennfat# cp f1 -h demo2copy
pennfat#
root@cis548Dev:~# cmp demo2 demo2copy
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 3 (Max Out)*:

```
root@cis548Dev:~# yes a | head -c 32000 > demo3
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# cp -h demo3 f2
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 4 (Overwrite)*:

```
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# cat -w f2
hello
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 5 (Expand Directory)*:

```
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# touch f3 f4 f5
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 6 (Append without New Block)*:

```
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# cat -a f2
hi
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 7 (Remove File)*:

```
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# rm f1
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 8 (File Not Found)*:

```
root@cis548Dev:~# pennfat
pennfat# mount minfs
```

```
pennfat# cat f1
...
pennfat# cp f1 f2
...
pennfat# mv f1 f2
...
pennfat# rm f1
...
pennfat#
root@cis548Dev:~#
```

- *Demo 9 (Append with New Blocks)*:

```
root@cis548Dev:~# head -c 256 /etc/legal > demo9
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# cp -h demo9 f1
pennfat# cat f1 -a f2
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 10 (Move Files)*:

```
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# mv f1 f5
pennfat# mv f2 f5
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 11 (Max Out, Non-contiguous)*:

```
root@cis548Dev:~# yes b | head -c 31488 > demo11
root@cis548Dev:~# pennfat
pennfat# mount minfs
pennfat# cp -h demo11 f1
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd minfs
...
root@cis548Dev:~#
```

- *Demo 12 (Unmount)*:

```
root@cis548Dev:~# pennfat
pennfat# mount maxfs
pennfat# cp -h demo3 f1
pennfat# ls
...
pennfat# umount
pennfat# mount minfs
pennfat# ls
...
pennfat#
root@cis548Dev:~# hd maxfs
...
root@cis548Dev:~#
```

3 Shell

Shell Testing

- *Demo 1*: Redirection.

Job Control and Built-Ins

- *Demo 1*: Show `fg`
- *Demo 2*: Show `bg`
- *Demo 3*: Show `sleep` in the foreground and check the log file.
- *Demo 4*: Show three different `sleep` in the background and check the log file.

```
$ sleep 9 &
[1] 2
$ sleep 8 &
[2] 3
$ sleep 7 &
[3] 4
$
[1]   Done         sleep 9
[2]   Done         sleep 8
[3]+  Done         sleep 7
$ ps
PID PPID PRI STAT CMD
  1    0  -1  B   shell
  5    1   0  R   ps
$
```

- *Demo 5*: Show all other required built-ins.
  - `cat`
  - `echo`
  - `ls`
  - `touch`
  - `mv`
  - `cp`
  - `rm`
  - `chmod`
  - `ps`
  - `kill`
  - `nice`
  - `nice_pid`
  - `man`
  - `jobs`
  - `logout`
- *Demo 6*: Show the functioning of shell scripts.

```
$ echo echo line1 > script
$ echo echo line2 >> script
$ cat script
echo line1
echo line2
$ chmod +x script
$ script > out
$ cat out
line1
line2
```

4 Submission and Documentation

- Good build system (header guards, good Makefile, subdirectories, etc.)
- Error handling
- Documentation

5 General Notes (Extra Credit, etc.):

Extra credit will be evaluated after the demo. Any EC must be thoroughly documented so that TAs can easily test and evaluate the code on their own.