**Qiman Wang**

**Homework # 2**

**10/7/17**

**PART A**

Root Directory (/):

qiman@qiman-ThinkPad-Yoga-14:/$ cd /

qiman@qiman-ThinkPad-Yoga-14:/$ ls -l

total 160

drwxr-xr-x   2 root root  4096 May 31 12:34 bin

drwxr-xr-x   4 root root  4096 May 31 12:39 boot

drwxrwxr-x   2 root root  4096 Nov 30  2016 cdrom

/bin

qiman@qiman-ThinkPad-Yoga-14:/$ cd bin

qiman@qiman-ThinkPad-Yoga-14:/bin$ ls -l

total 12976

-rwxr-xr-x 1 root root 1037528 May 16 08:49 bash

-rwxr-xr-x 1 root root   31288 May 20  2015 bunzip2

-rwxr-xr-x 1 root root 1964536 Aug 19  2015 busybox

List 6 commands I recongnize:

Cp,dir,echo,ls,mkdir,rmdir.

/dev

qiman@qiman-ThinkPad-Yoga-14:/bin$ cd /dev

qiman@qiman-ThinkPad-Yoga-14:/dev$ ls -l

total 0

crw-------  1 root root     10, 235 Oct  5 15:39 autofs

drwxr-xr-x  2 root root         340 Oct  5 11:39 block

drwxr-xr-x  2 root root          60 Oct  5 11:39 bsg

crw-------  1 root root     10, 234 Oct  5 15:39 btrfs-control

drwxr-xr-x  3 root root          60 Oct  5 11:39 bus

I recognize:

Core,cpu,disk,input,net,

/etc

qiman@qiman-ThinkPad-Yoga-14:/dev$ cd /etc

qiman@qiman-ThinkPad-Yoga-14:/etc$ ls -l

total 1184

drwxr-xr-x  3 root root     4096 Jul 19  2016 acpi

-rw-r--r--  1 root root     3028 Jul 19  2016 adduser.conf

-rw-r--r--  1 root root       16 Nov 30  2016 adjtime

drwxr-xr-x  2 root root    12288 May 31 12:37 alternatives

Files I’ve heard about:

Apt,emac,firefox,fonts,init,java-8-oracle,networkmanager,passwd,python,vim

It seems like the most used permission is r(read).

qiman@qiman-ThinkPad-Yoga-14:/etc$ cat passwd

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

sys:x:3:3:sys:/dev:/usr/sbin/nologin

sync:x:4:65534:sync:/bin:/bin/sync

I found myself:

qiman:x:1000:1000:Qiman Wang,,,:/home/qiman:/bin/bash

/lib

qiman@qiman-ThinkPad-Yoga-14:/$ cd /lib && ls -l

total 204

drwxr-xr-x  2 root root  4096 May 31 12:35 apparmor

drwxr-xr-x  2 root root  4096 Jul 19  2016 brltty

lrwxrwxrwx  1 root root    21 Nov 30  2016 cpp -> /etc/alternatives/cpp

drwxr-xr-x  3 root root  4096 Jul 19  2016 crda

/tmp

qiman@qiman-ThinkPad-Yoga-14:/lib$ cd /tmp && ls -l

total 24

-rw------- 1 root  root  10240 Oct  5 15:55 002f859e3b099

-rw------- 1 qiman qiman     0 Oct  5 15:39 config-err-0fnHmG

drwx------ 3 root  root   4096 Oct  5 15:39 systemd-private-9c1004829d784eba871164009f983e44-colord.service-iB6PHK

drwx------ 3 root  root   4096 Oct  5 15:39 systemd-private-9c1004829d784eba87116400

**PART B**

10.

Determine the absolute pathname for your home directory

qiman@qiman-ThinkPad-Yoga-14:/etc$ echo $HOME

/home/qiman

11.

qiman@qiman-ThinkPad-Yoga-14:/etc$ pwd

/etc

**PART C**

qiman@qiman-ThinkPad-Yoga-14:/etc$ echo $SHELL

/bin/bash

2. Available shells:

qiman@qiman-ThinkPad-Yoga-14:/etc$ cat /etc/shells

# /etc/shells: valid login shells

/bin/sh

/bin/dash

/bin/bash

/bin/rbash

/bin/csh

/bin/tcsh

/usr/bin/tcsh

5. After changing shell to tcsh, and ps to run processes:

qiman-ThinkPad-Yoga-14:~> ps

 PID TTY          TIME CMD

2132 pts/18   00:00:00 tcsh

3616 pts/18   00:00:00 ps

Shell environment variables:

qiman-ThinkPad-Yoga-14:/home> env

XDG\_SEAT\_PATH=/org/freedesktop/DisplayManager/Seat0

XDG\_CONFIG\_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg

LANG=en\_US.UTF-8

DISPLAY=:0

SHLVL=1

LOGNAME=qiman

qiman-ThinkPad-Yoga-14:/> pwd

/

**PART D. processes**

2.

PROCESS STATE CODES

      Here are the different values that the s, stat and state output

      specifiers (header "STAT" or "S") will display to describe the state of

      a process:

              D    uninterruptible sleep (usually IO)

              R    running or runnable (on run queue)

              S    interruptible sleep (waiting for an event to complete)

              T    stopped by job control signal

              t    stopped by debugger during the tracing

              W    paging (not valid since the 2.6.xx kernel)

              X    dead (should never be seen)

              Z    defunct ("zombie") process, terminated but not reaped by

                   its parent

 For BSD formats and when the stat keyword is used, additional

      characters may be displayed:

              <    high-priority (not nice to other users)

              N    low-priority (nice to other users)

              L    has pages locked into memory (for real-time and custom IO)

              s    is a session leader

              l    is multi-threaded (using CLONE\_THREAD, like NPTL pthreads

                   do)

              +    is in the foreground process group

3.

qiman-ThinkPad-Yoga-14:~/Desktop> ps -l

F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD

0 S  1000  2132  2123  0  80   0 -  5328 sigsus pts/18   00:00:00 tcsh

4 R  1000  6358  2132  0  80   0 -  7229 -      pts/18   00:00:00 ps

F: flags

S:process status code

PID:Process ID number

Ppid:id number of the process’s parent process

C:cpu usage and scheduling info

PRI:priority of the process

NI:nice value

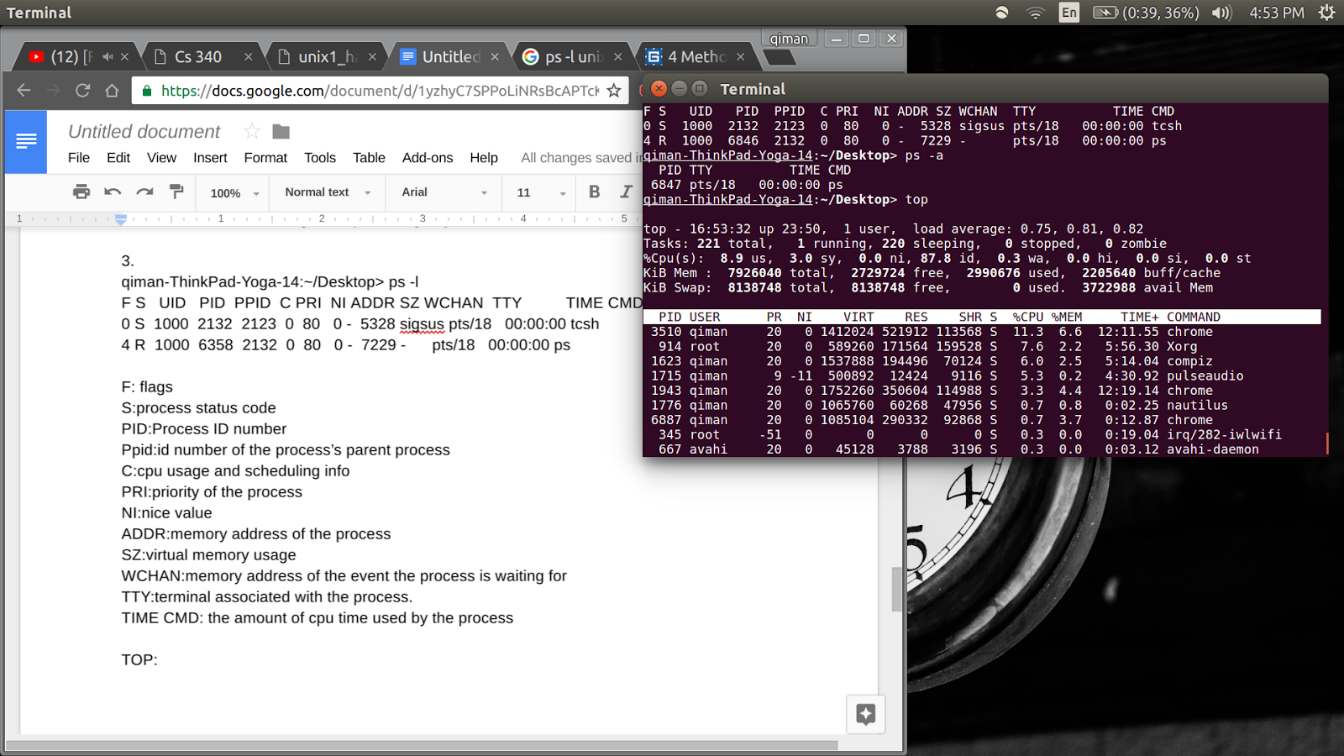
ADDR:memory address of the process

SZ:virtual memory usage

WCHAN:memory address of the event the process is waiting for

TTY:terminal associated with the process.

TIME CMD: the amount of cpu time used by the process

TOP:

Give the total number of tasks:218

number of running processes:1

sleeping processes:217

stopped processes:0

zombies:0

**PART E:**

1.The function CreateProcess() is used to create

A new process and a primary thready for the new process. The createProcess function calls the windows kernel functions  NtCreateProcess and NTCreateThread.

Return errorcode of 0 means failed, else it succeed.

The thread and process gets a unique id that last until it terminates. The unix fork() takes no parameter, while windows createprocess() takes in many. ExitProcess() is used to exit a process.

2.

qiman@qiman-ThinkPad-Yoga-14:~/Desktop/fall 2017/cs340$ ./parent

Process[3824]: Parent in execution …

// fork() is called and child process is created. Then the two process runs in parallel.

//parent then sleeps for 5 millisecond to wait for child process.

Process[3825]: child in execution …

//child process is created by parent’s fork()

//child then sleep for 1 millisecond

Process[3825]: child terminating …

//child terminates first because the sleep time is less than parent sleep time.

Process[3824]: Parent detects terminating child

//parent wakes up form sleep

//the parent’s wait() will receive a -1 which indicates child has terminated.

Process[3824]: Parent terminating …

//the parent then terminates also.

**OPRHAN**

qiman@qiman-ThinkPad-Yoga-14:~/Desktop/fall 2017/cs340$ ./orphan

I'm the original process with PID 3842 and PPID 3773.

//the process starts and creates a child process which should run concurrently

I'm the parent process with PID 3842 and PPID 3773.

my child's PID 3843

//fork() is called and it returns the PID of child to the parent process,

//return 0 to child process

//if failed, it will return -1 to the parent and no child is created.

//if PID is not zero, then it is the parent process.

//The parent print’s it’s id, it’s parent’s id and it’s child’s id

PID 3842 terminates.

//the parent terminates because all the other conditions is else.

qiman@qiman-ThinkPad-Yoga-14:~/Desktop/fall 2017/cs340$ I'm the child process with PID 3843 and PPID 1209.

//child sleeps for 5 millisecond, parent terminates first. The child’s return value is 0.

//child outputs it’s id and parent’s id.

//because the parent die before the child, the child is adopted by the original init process of PID=1. So the child prints it’s PID =1.

PID 3843 terminates.

//child terminate and the program ends