Rebecca Katz

CS 340- HW2

Dr. Fluture

**A. Standard Directories and Files:**

**0.1 List the Root Directory**

[kare6466@venus ~]$ ls -l

drwx------. 3 kare6466 underg 4096 Apr 29 2013 cs211

drwxr-xr-x. 2 kare6466 underg 4096 Jul 13 17:33 cs340

drwx------. 4 kare6466 underg 4096 Jan 28 2013 csci111

drwx------. 2 kare6466 underg 4096 Jul 13 17:41 exam1Review

drwx------. 2 kare6466 underg 4096 Dec 5 2012 exam2

drwx------. 2 kare6466 underg 4096 Dec 11 2012 finalReview

**0.2 Go to the bin Directory**

[kare6466@venus ~]$ cd /bin

**0.3 List the Contents of the bin Directory**

[kare6466@venus bin]$ ls -l

-rwxr-xr-x. 1 root root 123 Feb 21 2013 alsaunmute

-rwxr-xr-x. 1 root root 938832 Jul 18 2013 bash

-rwxr-xr-x. 1 root root 48568 Jun 25 04:55 cat

-rwxr-xr-x. 1 root root 59392 Jun 25 04:55 date

-rwxr-xr-x. 1 root root 49384 Jun 25 04:55 mkdir

-rwxr-xr-x. 1 root root 40480 Jun 25 04:55 rmdir

**0.4 List 6 Commands Recognized from the bin Directory Listing**

rmdir = remove directory, mkdir = make directory, date, cat, vi, view -> vi

**0.5 List the dev Directory and Recognized Commands**

[kare6466@venus bin]$ cd /dev

[kare6466@venus dev]$ ls -l

drwxr-xr-x. 3 root root 60 Jun 20 19:09 bus

lrwxrwxrwx. 1 root root 3 Jun 20 19:09 cdrom -> sr0

drwxr-xr-x. 2 root root 3400 Jun 20 19:10 char

crw-------. 1 root root 5, 1 Jun 20 19:09 console

drwxr-xr-x. 6 root root 140 Jun 20 19:09 cpu

brw-rw----. 1 root disk 1, 0 Jun 20 19:09 ram0

I recognize the ram, loop, cpu, and bus devices, cdrom, crash, etc.

**0.6 Go to the etc Directory**

[kare6466@venus dev]$ cd /etc

**0.7 Give a Long Listing and Mention Recognized Files**

[kare6466@venus etc]$ ls -l

-rw-r--r--. 1 root root 658 Jun 9 03:11 auto.master

-rw-r--r--. 1 root root 524 Jun 9 03:11 auto.misc

-rwxr-xr-x. 1 root root 1260 Jun 9 03:11 auto.net

-rwxr-xr-x. 1 root root 687 Jun 9 03:11 auto.smb

drwxr-xr-x. 2 root root 4096 Jun 17 2013 bluetooth

drwxr-xr-x. 2 root root 4096 Dec 4 2013 ssh

I recognize the following files: Bluetooth, Java, issue, environment, ssh.

**0.8 What is the Most Used Permission? What does it mean?**

**Drwxr-xr-x** Represents the permission settings:

**D** = directory entry

**rwx**= Owner of the file can: read, write, execute

**r-x** = Group members can: read, no access to write, execute

**r-x** = Others can: read, no access to write, execute

**0.9 Using cat, check the Passwd File**

[kare6466@venus etc]$ cat passwd

chko4431:x:5635:800:Kok-Chiu Cheuk:/home/su14/111/chko4431:/bin/bash

dial2184:x:5636:800:Alessandro DiMarco:/home/su14/111/dial2184:/bin/bash

escr6488:x:5638:800:Crital P Escobar:/home/su14/111/escr6488:/bin/bash

lebe2248:x:5640:800:Benjamin Lee:/home/su14/111/lebe2248:/bin/bash

lixi0021:x:5641:800:Xing Ying Li:/home/su14/111/lixi0021:/bin/bash

ahsa5873:x:5651:800:Sara Ahmad:/home/su14/340/ahsa5873:/bin/bash

Searching for my name did not return any results.

**B. Absolute pathname for home directory:**

**0.10 echo $HOME**

[kare6466@venus tmp]$ echo $HOME

/home/su14/340/kare6466

**0.11 pwd**

[kare6466@venus tmp]$ pwd

/tmp

[kare6466@venus tmp]$

**C. Shell and Shell Environment Variables:**

**1.1 Check Default Shell using echo $SHELL**

[kare6466@venus ~]$ echo $SHELL

/bin/bash

**1.2 Using chsh Find a List of Available Shells**

[kare6466@venus ~]$ chsh -l

/bin/sh

/bin/bash

/sbin/nologin

/bin/dash

/bin/tcsh

/bin/csh

/bin/ksh " does not exist.

**1.3 Change the Current shell to tcsh**

[kare6466@venus ~]$ chsh

Changing shell for kare6466.

Password:

New shell [/bin/bash]: /bin/tcsh

Shell changed.

**1.4 Check New Shell**

[kare6466@venus ~]$ echo $SHELL

/bin/tcsh

**1.5 Type ps and Mention What You Observe**

[kare6466@venus ~]$ ps

PID TTY TIME CMD

2811 pts/6 00:00:00 tcsh

2861 pts/6 00:00:00 ps

The process ID’s are listed and the time is 00:00:00

**1.6 Type “set | more”, What is Displayed?**

[kare6466@venus ~]$ set|more

COLORS /etc/DIR\_COLORS

\_ ps

addsuffix

argv ()

autolist

consoletype pty

csubstnonl

cwd /home/su14/340/kare6466

dirstack /home/su14/340/kare6466

**1.7 Identify and List the Settings for the Variable Shown Above**

file /home/su14/340/kare6466/.i18n

home /home/su14/340/kare6466

user kare6466

shell /bin/tcsh

**1.8 Using the man Command, Read About the setenv Command**

[kare6466@venus ~]$ man setenv

SETENV(3) Linux Programmerâs Manual SETENV(3)

NAME

setenv - change or add an environment variable

SYNOPSIS

#include <stdlib.h>

int setenv(const char \*name, const char \*value, int overwrite);

**1.9 Type csh, setenv | more. Identify and List the Settings for all Environment Variables.**

[kare6466@venus ~]$ setenv | more

USER=kare6466

LOGNAME=kare6466

HOME=/home/su14/340/kare6466

PATH=/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin

MAIL=/var/spool/mail/kare6466

SHELL=/bin/tcsh

SSH\_CLIENT=149.4.115.5 54667 22

SSH\_CONNECTION=149.4.115.5 54667 149.4.211.180 22

SSH\_TTY=/dev/pts/6

TERM=vt100

SELINUX\_ROLE\_REQUESTED=

SELINUX\_LEVEL\_REQUESTED=

SELINUX\_USE\_CURRENT\_RANGE=

HOSTTYPE=x86\_64-linux

VENDOR=unknown

OSTYPE=linux

MACHTYPE=x86\_64

SHLVL=2

PWD=/home/su14/340/kare6466

GROUP=underg

HOST=venus.cs.qc.cuny.edu

REMOTEHOST=bsc3.qc.cuny.edu

HOSTNAME=venus.cs.qc.cuny.edu

**D. Processes:**

**2.1 Learn about the ps Command Using man**

[kare6466@venus ~]$ man ps

PS(1) Linux Userâs Manual PS(1)

NAME

ps - report a snapshot of the current processes.

SYNOPSIS

ps [options]

DESCRIPTION

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use top(1) instead.

This version of ps accepts several kinds of options:

1 UNIX options, which may be grouped and must be preceded by a dash.

2 BSD options, which may be grouped and must not be used with a dash.

3 GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and ps implementations that this ps is compatible with.

**2.2 List Possible System States and Their Significance. Identify Login Shell.**

D  =  Uninterruptible sleep (usually IO)

R  = Running or runnable (on run queue)

S  = Interruptible sleep (waiting for an event to complete)

T  = Stopped, either by a job control signal or because it’s being traced.

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

X   =  dead (should never be seen)

Z   =   Zombie process, terminated but not reaped by its parent.

**2.3 ps -l**

F = process flags

S = process state

UID = user ID number of caller

PID = process ID

PPID = parent process ID

C = processor usage over the lifetime of the process

PRI = kernel scheduling priority

NI = nice value, ranging from 19 to ­20 and affects scheduling priority

ADDR = memory address of the process

SZ = size in physical pages of the core image of the process

WCHAN = address of the kernel function where the process is sleeping

TTY = terminal number

TIME = cumulative execution time

CMD = name of the executable

**2.4 The top Command Displays the Status of the first 15 Most CPU-Intensive Tasks and Their Activity on the System**

[kare6466@venus ~]$ top

top - 12:39:17 up 24 days, 17:29, 8 users, load average: 0.00, 0.00, 0.00

**2.5 Tasks: 311 total, 1 running, 310 sleeping, 0 stopped, 0 zombie**

Cpu(s): 0.1%us, 0.3%sy, 0.0%ni, 99.6%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st

Mem: 65897984k total, 44770728k used, 21127256k free, 583132k buffers

Swap: 12287992k total, 0k used, 12287992k free, 42189136k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

9656 oracle -2 0 19.2g 17m 15m S 1.7 0.0 526:39.10 ora\_vktm\_csdept

5113 kare6466 20 0 15168 1404 944 R 0.3 0.0 0:00.21 top

9676 oracle 20 0 19.2g 132m 129m S 0.3 0.2 7:42.82 ora\_ckpt\_csdept

1 root 20 0 19356 1460 1144 S 0.0 0.0 0:41.85 init

2 root 20 0 0 0 0 S 0.0 0.0 0:00.06 kthreadd

3 root RT 0 0 0 0 S 0.0 0.0 0:36.62 migration/0

4 root 20 0 0 0 0 S 0.0 0.0 0:21.22 ksoftirqd/0

**E. Commands Used in the Creation/ Termination of Processes**

**3.1 fork()** creates a new process, the child, by duplicating the calling process, the parent.

**execve()** executes the program pointed to by filename.

**wait()** wait for a child process to terminate or stop, and determine its status.

**kill()** sends a signal to a specified process or process group. Processes which do not catch this signal are killed.

**3.2 Give an Overview of the Command Used in Windows for Creating Processes**

Windows has a *CreateProcess* function to create new processes and primary threads. The new process runs in the security context of the calling process. If the calling process is impersonating another user, the new process uses the token for the calling process, not the impersonation token. To run the new process in the security context of the user represented by the impersonation token, use the *CreateProcessAsUser* or *CreateProcessWithLogonW* function.

**Source:** http://msdn.microsoft.com/en-us/library/ms682425(VS.85).aspx

**3.3 Execute the Files**

**[kare6466@venus ~]$ ./parent**

Process[3180]: Parent in execution ...

Process[3181]: child in execution ...

Process[3181]: child terminating ...

Process[3180]: Parent detects terminating child

Process[3180]: Parent terminating ...

**[kare6466@venus ~]$ ./orphan**

I'm the original process with PID 3186 and PPID 2027.

I'm the parent process with PID 3186 and PPID 2027.

my child's PID 3187

PID 3186 terminates.

**3.4 Explain the Execution of the Programs**

The parent class creates two processes using the fork command and starts the child process. Process one, the parent, is executed then waits for the second process, the child, to finish executing. The child is executed and terminates using the execve command. Then the child process ID is given to the parent, which continues execution and the child process is now in the zombie state. The parent process terminates.

The orphan program runs similarly to the parent except in this case the parent process executes after the child process is created and doesn’t wait for the child to finish executing. In doing so, the program creates an “orphaned” child process, meaning the parent process terminated without receiving the value from the child process.