Name : Chia-Sheng Hsiao

ID : 23399066

Part A

1.

[hsch9066@venus ~]$ cd

[hsch9066@venus ~]$ ls -l

Display:

total 204

-rw-r--r-- 1 hsch9066 underg 95364 Apr 26 01:50 12.sol

drwxr-xr-x 29 hsch9066 underg 4096 May 22 2016 cs111

drwxr-xr-x 15 hsch9066 underg 4096 Jul 30 2016 cs211

-rw-r--r-- 1 hsch9066 underg 663 Nov 2 2007 CS316ex0.java

-rw-r--r-- 1 hsch9066 underg 1653 Nov 2 2007 CS316ex10.java

2.

[hsch9066@venus ~]$ cd /bin

3.

[hsch9066@venus bin]$ ls -l

Display:

-rwxr-xr-x. 1 root root 2859 Jul 27 2015 zmore

-rwxr-xr-x. 1 root root 5343 Jul 27 2015 znew

lrwxrwxrwx. 1 root root 6 Dec 12 2016 zsoelim -> soelim

-rwxr-xr-x 1 root root 4930352 Jun 10 02:41 zts-php

-rwxr-xr-x 1 root root 5560 Jun 10 02:41 zts-php-config

-rwxr-xr-x 1 root root 4788 Jun 10 02:41 zts-phpize

4.

List 6 commands that you recognize : cut, rmdir, mail, echo, date, mkdir

5.

[hsch9066@venus bin]$ cd /dev

[hsch9066@venus dev]$ ls -l

Display :

crw------- 1 root root 10, 63 Jun 5 12:12 vga\_arbiter

crw------- 1 root root 10, 137 Jun 5 12:12 vhci

crw------- 1 root root 10, 238 Jun 5 12:12 vhost-net

crw------- 1 root root 10, 130 Jun 5 12:12 watchdog

crw------- 1 root root 252, 0 Jun 5 12:12 watchdog0

crw-rw-rw- 1 root root 1, 5 Jun 5 12:12 zero

Devices I recognize are : Input, Disk, Log, Net, Ram, Mapper

6.

[hsch9066@venus dev]$ cd /etc

7.

[hsch9066@venus etc]$ ls -l

Display :

drwxr-xr-x. 5 root root 94 Nov 5 2016 xdg

drwxr-xr-x. 2 root root 6 Nov 5 2016 xinetd.d

drwxr-xr-x. 2 root root 20 Dec 12 2016 xml

drwxr-xr-x. 6 root root 4096 Dec 12 2016 yum

-rw-r--r--. 1 root root 970 Nov 15 2016 yum.conf

drwxr-xr-x. 2 root root 4096 Jan 11 2017 yum.repos.d

Files I’ve heard : bluetooth, Java, diskdump, firmware, issue, mail

8.

The most used permission is: -rw-r--r—

rw : the most used permission for owner , it means “read” and “write” available.

r : the most used permission for group, it means “read” only available.

r (the second one) : the most used permission for other, it means “read” only

available.

9.

(1)

[hsch9066@venus etc]$ cat profile

Display :

# /etc/profile

# System wide environment and startup programs, for login setup

# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you

# are doing. It's much better to create a custom.sh shell script in

# /etc/profile.d/ to make custom changes to your environment, as this

# will prevent the need for merging in future updates.

pathmunge () {

case ":${PATH}:" in

(2)

[hsch9066@venus etc]$ cat login.defs

Display :

#

# Please note that the parameters in this configuration file control the

# behavior of the tools from the shadow-utils component. None of these

# tools uses the PAM mechanism, and the utilities that use PAM (such as the

# passwd command) should therefore be configured elsewhere. Refer to

# /etc/pam.d/system-auth for more information.

#

# \*REQUIRED\*

# Directory where mailboxes reside, \_or\_ name of file, relative to the

# home directory. If you \_do\_ define both, MAIL\_DIR takes precedence.

# QMAIL\_DIR is for Qmail

#

#QMAIL\_DIR Maildir

MAIL\_DIR /var/spool/mail

10.

[hsch9066@venus etc]$ cat passwd |more

Display :

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

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

my file :

hsch9066:x:1801:800:Chia-Sheng Hsiao:/home/su17/340/hsch9066:/bin/bash

Part B

11.

[hsch9066@venus etc]$ echo $HOME

Display :

/home/su17/340/hsch9066

12.

[hsch9066@venus etc]$ pwd

Display :

/etc

Part C

1.

[hsch9066@venus ~]$ echo $SHELL

Display :

/bin/bash

2.

[hsch9066@venus ~]$ chsh -l

Display :

/bin/sh

/bin/bash

/sbin/nologin

/usr/bin/sh

/usr/bin/bash

/usr/sbin/nologin

/bin/tcsh

/bin/csh

/bin/ksh

/bin/rksh

3.

[hsch9066@venus ~]$ chsh

Changing shell for hsch9066.

New shell [/bin/bash]:

Password:

Shell changed.

4.

[hsch9066@venus ~]$ echo $SHELL

Display :

/bin/tcsh

5.

[hsch9066@venus ~]$ ps

Display :

PID TTY TIME CMD

92206 pts/10 00:00:00 tcsh

100278 pts/10 00:00:00 ps

Part D

1.

[hsch9066@venus ~]$ man ps

Display :

PS(1) User Commands PS(1)

NAME

ps - report a snapshot of the current processes.

SYNOPSIS

ps [options]

DESCRIPTION

2.

[hsch9066@venus ~]$ man ps

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.

[hsch9066@venus ~]$ ps -l

Display :

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

0 S 1801 92206 92203 0 80 0 - 29150 wait pts/10 00:00:00 bash

0 R 1801 102201 92206 0 80 0 - 37233 - pts/10 00:00:00 ps

F – Flag of peocess

S – State of Process

UID – User ID

PID – Process ID

PPID – Parent Process ID

C – CPU’s use of the process

PRI – Priority of process

NI – Used to get PRI

ADDR – memory address of the process

SZ – Size of the process

WCHAN – The event that the process is waiting for

TTY – the name of the console that the user is logged into

TIME – the amount of CPU in minutes and seconds that the process has been running

CMD – the name of the command that launched the process

Internet source : https://shapeshed.com/unix-ps/

4.

[hsch9066@venus ~]$ top

Display :

top - 22:52:06 up 41 days, 10:39, 18 users, load average: 1.00, 1.03, 1.05

Tasks: 1326 total, 2 running, 1324 sleeping, 0 stopped, 0 zombie

%Cpu(s): 0.8 us, 0.0 sy, 0.0 ni, 99.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 65748024 total, 2752184 free, 2515516 used, 60480324 buff/cache

KiB Swap: 32964604 total, 32875324 free, 89280 used. 49854012 avail Mem

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

49562 leda1209 20 0 119476 2016 1360 R 99.3 0.0 581:35.58 tcsh

102858 hsch9066 20 0 158960 3536 1568 R 1.6 0.0 0:00.53 top

10 root 20 0 0 0 0 S 0.3 0.0 41:28.07 rcu\_sched

8480 oracle 20 0 19.150g 23344 19396 S 0.3 0.0 51:00.41 ora\_dia0\_venus

80976 root 20 0 0 0 0 S 0.3 0.0 0:01.42 kworker/42:2

5.

Tasks: 1326 total

Running Process: 2

Sleeping Process: 1324

Stopped Process: 0

Zombie Process: 0

Part E

1.

CreateProcess function

Creates a new process and its primary thread. 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.

Internet source : https://msdn.microsoft.com/en-us/library/ms682425(VS.85).aspx

2.

[hsch9066@venus ~]$ ./parent

Process[108991]: Parent in execution ...

Process[108992]: child in execution ...

Process[108992]: child terminating ...

Process[108991]: Parent detects terminating child

Process[108991]: Parent terminating ...

[hsch9066@venus ~]$ ./orphan

I'm the original process with PID 109132 and PPID 92206.

I'm the parent process with PID 109132 and PPID 92206.

my child's PID 109133

PID 109132 terminates.

[hsch9066@venus ~]$ I'm the child process with PID 109133 and PPID 1.

PID 109133 terminates.

Parent runs, then child is allowed to run, parent waits while child executes and parent terminates after child terminates. Second command with “./orphan”, the parent terminates, then the child terminates with its exiting command as it is an orphan.

First, the parent class creates 2 process by using fork(), the first one is parent process, the second process is child process. The parent process is executed and then waits for the child process. After the child process is executed, gets process ID and terminates, the parent process continues to finish and terminates.

The orphan class is similarly to the parent class, but its parent process doesn’t wait for child process. It means that the parent process will be terminated without receiving information from child process.