Bronson Edralin

EE468 Operating Systems

HW #3

9/17/14

***Textbook Problem 3.10 (1 pt)*** *Next you are to construct a process tree similar to Figure 3.8 (do it on wiliki. Actually, create a tree using pstree. You can store it in a file using redirection: pstree > hw3tree. Print it out and turn it in.*

*Note that just using pstree may not be easy to read. So use the –A option to print it using ASCII. You can also use the –p option to print the PIDs as well. For other options see*

[*http://www.tutorialspoint.com/unix\_commands/pstree.htm*](http://www.tutorialspoint.com/unix_commands/pstree.htm)

**pstree –A > hw3tree**

init-+-acpid

|-apcupsd---2\*[{apcupsd}]

|-atd

|-auditd-+-audispd---{audispd}

| `-{auditd}

|-automount---4\*[{automount}]

|-avahi-daemon---avahi-daemon

|-cdsMsgServer

|-cdsNameServer

|-cdsXvnc

|-clsbd---{clsbd}

|-crond

|-cupsd

|-dbus-daemon---{dbus-daemon}

|-dhclient

|-2\*[dnsmasq]

|-events/0

|-events/1

|-events/2

|-events/3

|-events/4

|-events/5

|-events/6

|-events/7

|-exim

|-gam\_server

|-gdm-binary---gdm-binary-+-Xorg

| `-gdmgreeter

|-gdm-rh-security

|-gpm

|-hald---hald-runner-+-hald-addon-acpi

| |-2\*[hald-addon-keyb]

| `-hald-addon-stor

|-hcid

|-hidd

|-hpiod

|-hpssd.py

|-hv\_kvp\_daemon

|-irqbalance

|-2\*[iscsid]

|-iscsiuio---2\*[{iscsiuio}]

|-khelper

|-klogd

|-krfcommd

|-ksoftirqd/0

|-ksoftirqd/1

|-ksoftirqd/2

|-ksoftirqd/3

|-ksoftirqd/4

|-ksoftirqd/5

|-ksoftirqd/6

|-ksoftirqd/7

|-kthread-+-aio/0

| |-aio/1

| |-aio/2

| |-aio/3

| |-aio/4

| |-aio/5

| |-aio/6

| |-aio/7

| |-ata/0

| |-ata/1

| |-ata/2

| |-ata/3

| |-ata/4

| |-ata/5

| |-ata/6

| |-ata/7

| |-ata\_aux

| |-bnx2i\_thread/0

| |-bnx2i\_thread/1

| |-bnx2i\_thread/2

| |-bnx2i\_thread/3

| |-bnx2i\_thread/4

| |-bnx2i\_thread/5

| |-bnx2i\_thread/6

| |-bnx2i\_thread/7

| |-cnic\_wq

| |-cqueue/0

| |-cqueue/1

| |-cqueue/2

| |-cqueue/3

| |-cqueue/4

| |-cqueue/5

| |-cqueue/6

| |-cqueue/7

| |-ib\_addr

| |-ib\_cm/0

| |-ib\_cm/1

| |-ib\_cm/2

| |-ib\_cm/3

| |-ib\_cm/4

| |-ib\_cm/5

| |-ib\_cm/6

| |-ib\_cm/7

| |-ib\_inform

| |-ib\_mcast

| |-iscsi\_eh

| |-iw\_cm\_wq

| |-kacpid

| |-kauditd

| |-kblockd/0

| |-kblockd/1

| |-kblockd/1

| |-kblockd/2

| |-kblockd/3

| |-kblockd/4

| |-kblockd/5

| |-kblockd/6

| |-kblockd/7

| |-kedac

| |-khubd

| |-khungtaskd

| |-3\*[kjournald]

| |-kksmd

| |-kmpath\_handlerd

| |-kmpathd/0

| |-kmpathd/1

| |-kmpathd/2

| |-kmpathd/3

| |-kmpathd/4

| |-kmpathd/5

| |-kmpathd/6

| |-kmpathd/7

| |-kpsmoused

| |-kseriod

| |-ksnapd

| |-kstriped

| |-kswapd0

| |-local\_sa

| |-md0\_raid1

| |-md1\_raid1

| |-md2\_raid1

| |-mpt/0

| |-mpt\_poll\_0

| |-2\*[pdflush]

| |-rdma\_cm

| |-rpciod/0

| |-rpciod/1

| |-rpciod/2

| |-rpciod/3

| |-rpciod/4

| |-rpciod/5

| |-rpciod/6

| |-rpciod/7

| |-scsi\_eh\_0

| |-scsi\_eh\_1

| `-scsi\_eh\_2

|-libvirtd---6\*[{libvirtd}]

|-mcstransd

|-mdadm

|-migration/0

|-migration/1

|-migration/2

|-migration/3

|-migration/4

|-migration/5

|-migration/6

|-migration/7

|-6\*[mingetty]

|-ntpd

|-oaFSLockD

|-pcscd---{pcscd}

|-portmap

|-python

|-restorecond

|-rpc.idmapd

|-rpc.statd

|-sdpd

|-smartd

|-sshd-+-sshd---sshd---bash---virtuoso-+-cdsServIpc---ipvs---4\*[{ipvs}]

| | |-cdsServIpc---tclsh---{tclsh}

| | |-cdsServIpc---libManager

| | |-cdsServIpc---runICRP0---virtuoso-+-cdsServIpc---tclsh---{tclsh}

| | | `-cdsServIpc

| | `-2\*[{virtuoso}]

| |-sshd---sshd---bash---make---sh---vivado---loader---vivado---72\*[{vivado}]

| |-sshd---sshd-+-bash

| | `-sftp-server

| |-2\*[sshd---sshd---sftp-server]

| |-sshd---sshd---bash---pstree

| |-sshd---sshd---bash

| `-sshd---sshd---bash---top

|-syslogd

|-udevd

|-watchdog/0

|-watchdog/1

|-watchdog/2

|-watchdog/3

|-watchdog/4

|-watchdog/5

|-watchdog/6

|-watchdog/7

|-xfs

`-yum-updatesd

***Textbook Problem 3.2 (1 pt).*** *Include the initial parent process, how many process are created by the program in Figure 3.28 (below)?*

*#include <stdio.h>*

*#include <unistd.h>*

*int main()*

*{*

*/\* fork a child process \*/*

*fork();*

*/\* fork another child process \*/*

*fork();*

*/\* and fork another \*/*

*fork();*

*return 0;*

*}*

*Figure 3.28.*

**There are 8 processes. A fork occurred for child and parent. It just kept happening because there were no conditionals for parent and child.**

***Text Prob 3.1 (1 pt).*** *Using the program shown in Figure 3.30 (below), explain what the output will be at line A.*

*int value = 5;*

*int main()*

*{*

*pid\_t pid;*

*pid = fork();*

*if (pid == 0) { /\* child process \*/*

*value += 15;*

*return 0;*

*}*

*else if (pid > 0) { /\* parent process \*/*

*wait(NULL);*

*printf(“PARENT: value = %d, value); /\* Line A, Parent doesn’t change “value” = 5 \*/*

*return 0;*

*}*

*}*

*Figure 3.30.*

**The parent and child were 2 separate processes. The value was altered in the child, but the parent never saw the change because it was in a different process. The only way we could do this was by using pointers.**

***Problem A (2 pts).*** *Attached you will find a file shm\_server.c. This program creates a shared memory of 50 bytes with the name “5678”. Initially, it loads “Aloha” in the shared memory, terminated by the byte ‘\0’. Then it periodically (every second) checks if the first byte is the letter ‘t’. If it is then it deletes the shared memory and terminates. To run the program in background you enter*

*./server &*

*where “server” is the name of the executable.*

*You are to write a client program called shm\_client.c. This program will have a single word as an in-line argument. If the executable is “server” then you can run the program as follows:*

*./server hello*

*where “hello” is an arbitrary word. The program will do the following:*

* *Check if exactly one word is used as an argument, otherwise this is an error, it prints a message, and then terminates*
* *Looks for the shared memory “5678”. If it can’t then it prints an error message and terminates*
* *Attach the shared memory to a character string pointer. Otherwise, it prints an error message and terminates.*
* *Prints the current contents of shared memory onto the console. The contents are assumed to be terminated by ‘\0’. The program should print “Client: current shared memory = <the contents>”. Note that initially <the contents> = “Aloha”.*
* *Stores the word entered as an inline argument into the shared memory. In our example above, we would store “hello” into shared memory terminated by ‘\0’.*
* *Prints the contents of shared memory again onto the console. In the example above, this would be “Client: new shared memory = hello”*
* *Terminate*

*Your program should run on Ubuntu. Hint: I copied then modified the server program to make the client.*

***Submission instructions:***

* *Step 1: Upload shm\_server.c into laulima*
* *Step 2. Turn in the rest of the problems as hard copies*

#include <sys/types.h> /\* for shared memory \*/

#include <sys/ipc.h> /\* for shared memory \*/

#include <sys/shm.h> /\* for shared memory \*/

#include <stdio.h>

#include <stdlib.h> /\* for exit() \*/

#include <unistd.h> /\* for sleep() \*/

#include <string.h>

#define SHMSZ 50 /\* Shared memory size \*/

#define NUMSEC 60 /\* 60 seconds of lifetime \*/

#define SHMNAME 5678 /\* Name for the shared memory \*/

int main(int argc, char \*argv[]) {

int shmid;

key\_t key;

char \*shm, \*s;

key = 5678; // We need to get the segment named "5678", created by the server

if (argc != 2) {

printf("\nError: You must have only 1 ARGUMENT!\n\n");

exit(1);

}

else {

if ((shmid = shmget(key, SHMSZ, 0666)) < 0) { // Locate the segment

perror("shmget");

exit(1);

}

if ((shm = shmat(shmid, (void \*)0, 0)) == ((char \*)-1)) { // We attach segment

printf("\nShared Memory not successfully attached...\n\n");

perror("shmat");

exit(1);

}

printf("\nShared Memory successfully attached...\n");

printf("Current Shared Memory = %s\n",shm);

strncpy(shm, argv[1], SHMSZ); // Replace old shared memory

printf("New shared memory = %s\n\n",shm);

}

return 0;

}