CS4328: Homework #1

Due on Feb, 9, 2024 @ 11:55 pm (submission on Canvas)

Aaron Luna

PLEASE READ: You may discuss this problem set with other students. However, you must *write up your answers on your own*. You must also write the names of other students you discussed any problem with. Each problem has a different weight. Please state any assumptions you are making in solving a given problem. *Show your work*. Late assignments will not be accepted with prior arrangements. By submitting this assignment, you acknowledge that you have read the course syllabus.

Problem 1

A network card on a machine is receiving packets at an average rate of 100 packets per second. If we assume that each packet causes the network Interrupt Service Routine (ISR) to execute for 0.2 millisecond, what percentage of the CPU time is used in handling network packets? [**4 pts**]

100(0.2) = 20 millisecond $\qquad$ % CPU = $\left(\frac{\text{Time required}}{\text{total time}}\right) \times 100\%$

$\frac{20 \text{ millisecond}}{100} = 0.02(100) = 2\ \%$

Problem 2

Some architectures are designed so that processors have multiple register sets. Describe what happens when a context switch occurs if the new context (of the next process selected by the scheduler) is already loaded into one of the register sets. What happens if the new context is in memory rather than in a register set and all the register sets are in use? [**4 pts**]

A context switch simply requires changing the pointer to the current register set, context switch can be expedited if new context is already loaded into one available register sets. Required data and state are already loaded into registers, no need to save current state to memory or load new state.

context switch time is pure overhead since system does no useful work. It is typically faster between threads than between processes. Takes longer if stored in memory and register sets are in use. Current state of one of register sets must be saved to memory. Once freed up, new context must be loaded from memory into register set.

Problem 3

Consider a program that performs the following steps repeatedly:

1. Use the CPU for 4 milliseconds.
2. Issue in I/O to disk for 14 milliseconds.
3. Use the CPU for 10 milliseconds.
4. Issue an I/O to the network for 18 milliseconds.

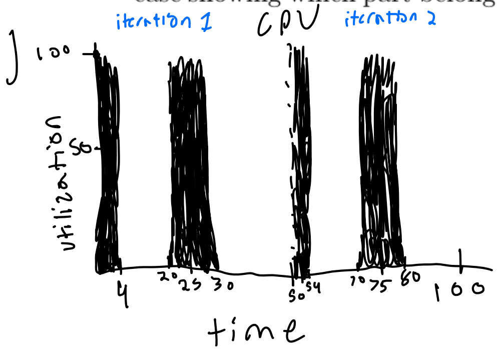
Assume that each step depends on data obtained from the previous step. Also, assume that it takes 2 millisecond to *execute* the device driver code on the CPU for both the disk and the network at the beginning of the I/O. Answer the following questions:

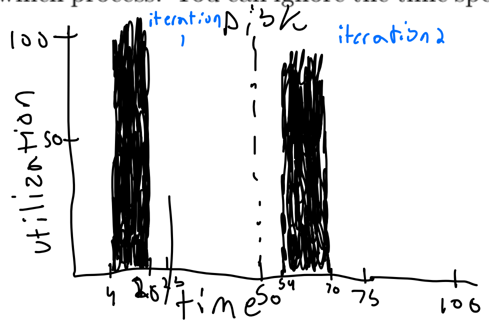
(a) Draw 3 time-line diagrams (time on the x-axis and utilization on the y-axis) that illustrate the utilizations of the CPU, disk, and network over the execution of two iterations of the program above. **[3 pts]**

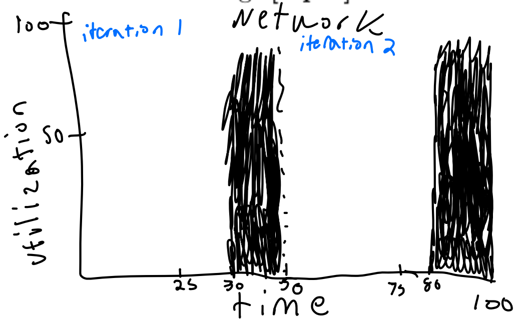
(b) What are the average utilizations of the CPU, disk and network over these two iterations? **[3 pts]**

(c & d) Assume that there are two processes of the program above running in a multiprogramming system (i.e., when a process blocks for I/O, another process can get the CPU), answer parts (a) and (b) for this case showing which part belong to which process. You can ignore the time spent in context switching. **[6 pts]**

a)





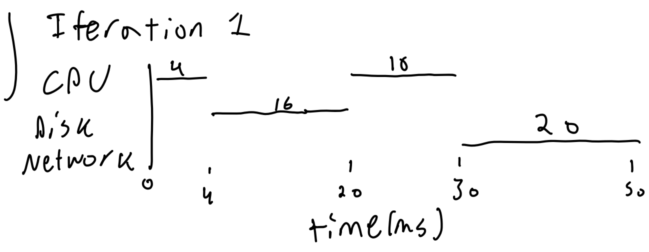


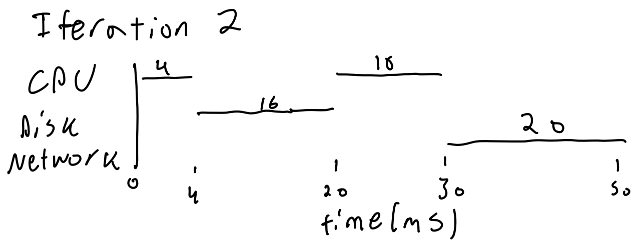
b) CPU = 4+10+4+10 /50 = 56.67/2 = 0.28ms

Disk = 2+14+2+14 /50 = 64.33/2 = 0.32ms

Network = 2+18+2+18 /50 = 80.67/2 = 0.4 ms

c)





d) CPU = 10+4+10+4/100 = 0.28/2 = 0.14 ms

Disk = 2+14+2+14/100 = 0.32/2 = 0.16 ms

Network = 2+18+2+18/100 = 0.4/2 = 0.2 ms

Problem 4

Consider a multiprogramming system with 4 processes, A, B, C, and D that arrived at the same time. Processes A and B are CPU-bound ones, while processes C and D are I/O-bound ones. Their resources requirements are given in the table below. Assume that process A uses 80% of the CPU while running, process B uses 80% of the CPU while running, process C uses 10% of the CPU while executing I/O, and process D uses 10% of the CPU while executing I/O. Assume processes C and D use their respective I/O devices 90% of the time while executing their I/O. The total memory available in the system to be used for all jobs is 250MB (assume no virtual memory).

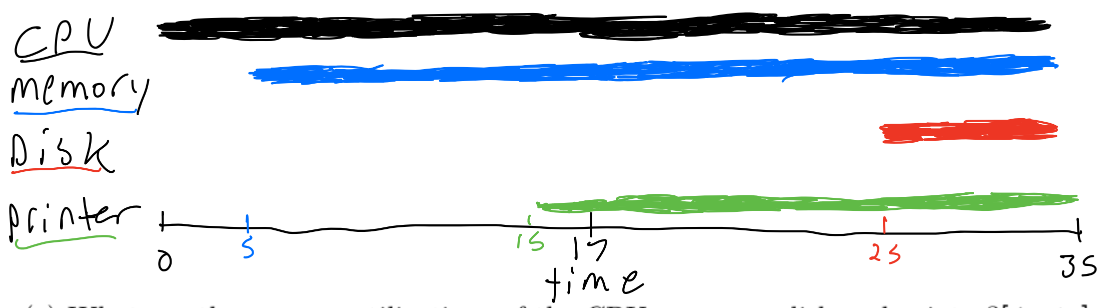
	Process A	Process B	Process C	Process D
Duration	5 min	15 min	5 min	10 min
Memory	40M	50M	150M	100M
Need Disk	No	No	Yes	Yes
Need printer	No	No	No	Yes

Answer the following questions:

(a) How long would it take until all the processes are done?[**4 pts**]

since it is a multiprogramming system, all processes would be done simultaneously. Therefore all will be done after 15 minutes

(b) Draw the utilization of the CPU, memory, Disk and printer over time?[**4 pts**]



(c) What are the average utilizations of the CPU, memory, disk and printer?[**4 pts**]

CPU utilization $= (0.8 + 0.8 + 0.1 + 0.1)/4 = 0.45 \times 100 = 45\%$
Mem utilization $= (40 + 50 + 150 + 100)/4 = 85$ M
Disk utilization $= 2/4 = 0.5 \times 100 = 50\%$
printer utilization $= 1/4 = 0.25 \times 100 = 25\%$

(d) What is the overall system throughput (in jobs/hr)?[**4 pts**]

4 processes @ 15 min $= (4/15)$
$60/15 = 4$, $(4/15)\,4 = 16/60$ min
$= 16$ processes $/ 1$ hour

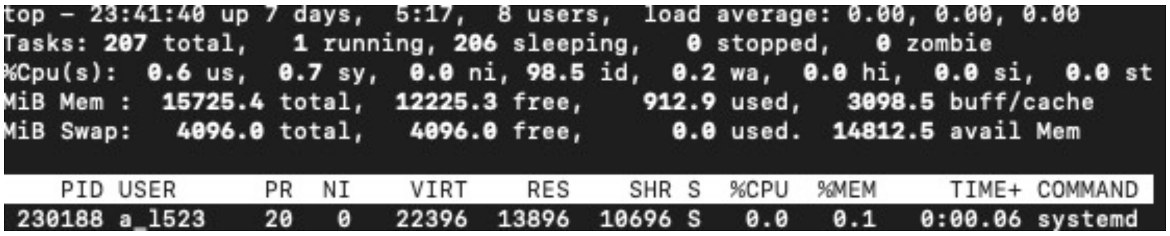
(e) What is the average turnaround time for all processes?[**4 pts**]

$$\frac{5+15+5+10}{4} = \frac{35}{4} = 8.75$$ = 8 minutes 45 seconds

Problem 5

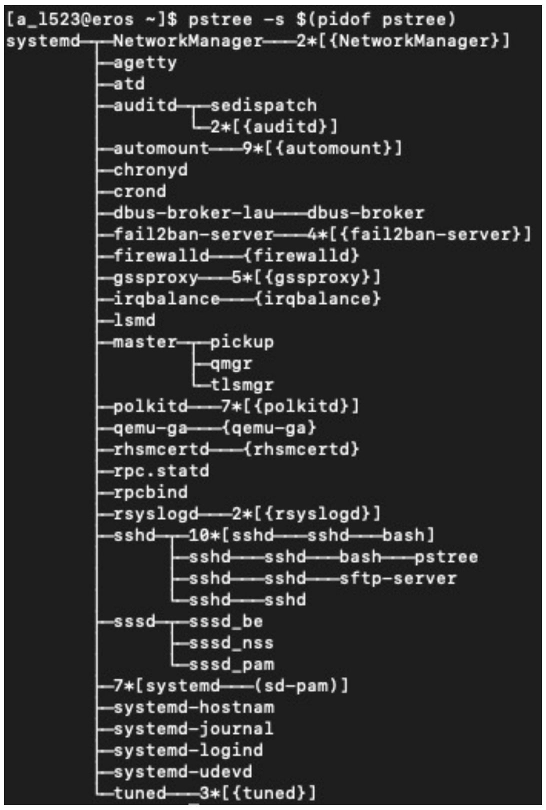
Log into eros.cs.txstate.edu using "ssh" and answer the following questions: **[2 pts each]**

(a) Run "top" with appropriate arguments and determine the memory utilization.



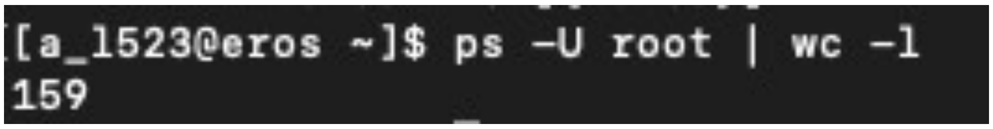
```
top - 23:41:40 up 7 days,  5:17,  8 users,  load average: 0.00, 0.00, 0.00
Tasks: 207 total,   1 running, 206 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.6 us,  0.7 sy,  0.0 ni, 98.5 id,  0.2 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  15725.4 total,  12225.3 free,    912.9 used,   3098.5 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.  14812.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 230188 a_1523    20   0   22396  13896  10696 S   0.0   0.1   0:00.06 systemd
```

(b) Run "pstree" with appropriate arguments and show recursively its (i.e., the pstree process) complete list of parents.



```
[a_1523@eros ~]$ pstree -s $(pidof pstree)
systemd─┬─NetworkManager───2*[{NetworkManager}]
        ├─agetty
        ├─atd
        ├─auditd─┬─sedispatch
        │        └─2*[{auditd}]
        ├─automount───9*[{automount}]
        ├─chronyd
        ├─crond
        ├─dbus-broker-lau───dbus-broker
        ├─fail2ban-server───4*[{fail2ban-server}]
        ├─firewalld───{firewalld}
        ├─gssproxy───5*[{gssproxy}]
        ├─irqbalance───{irqbalance}
        ├─lsmd
        ├─master─┬─pickup
        │        ├─qmgr
        │        └─tlsmgr
        ├─polkitd───7*[{polkitd}]
        ├─qemu-ga───{qemu-ga}
        ├─rhsmcertd───{rhsmcertd}
        ├─rpc.statd
        ├─rpcbind
        ├─rsyslogd───2*[{rsyslogd}]
        ├─sshd─┬─10*[sshd───sshd───bash]
        │      ├─sshd───sshd───bash───pstree
        │      ├─sshd───sshd───sftp-server
        │      └─sshd───sshd
        ├─sssd─┬─sssd_be
        │      ├─sssd_nss
        │      └─sssd_pam
        ├─7*[systemd───(sd-pam)]
        ├─systemd-hostnam
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-udevd
        └─tuned───3*[{tuned}]
```

(c) Run "ps" with appropriate arguments and show how many processes belong to the "root" user.



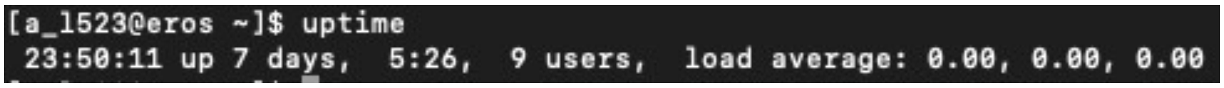
```
[a_1523@eros ~]$ ps -U root | wc -l
159
```

159

(d) What command would you use to know for how long the server was up? How long was that time since the machine was last powered on?

For this problem, you are allowed to research the correct parameters/commands to use.

uptime

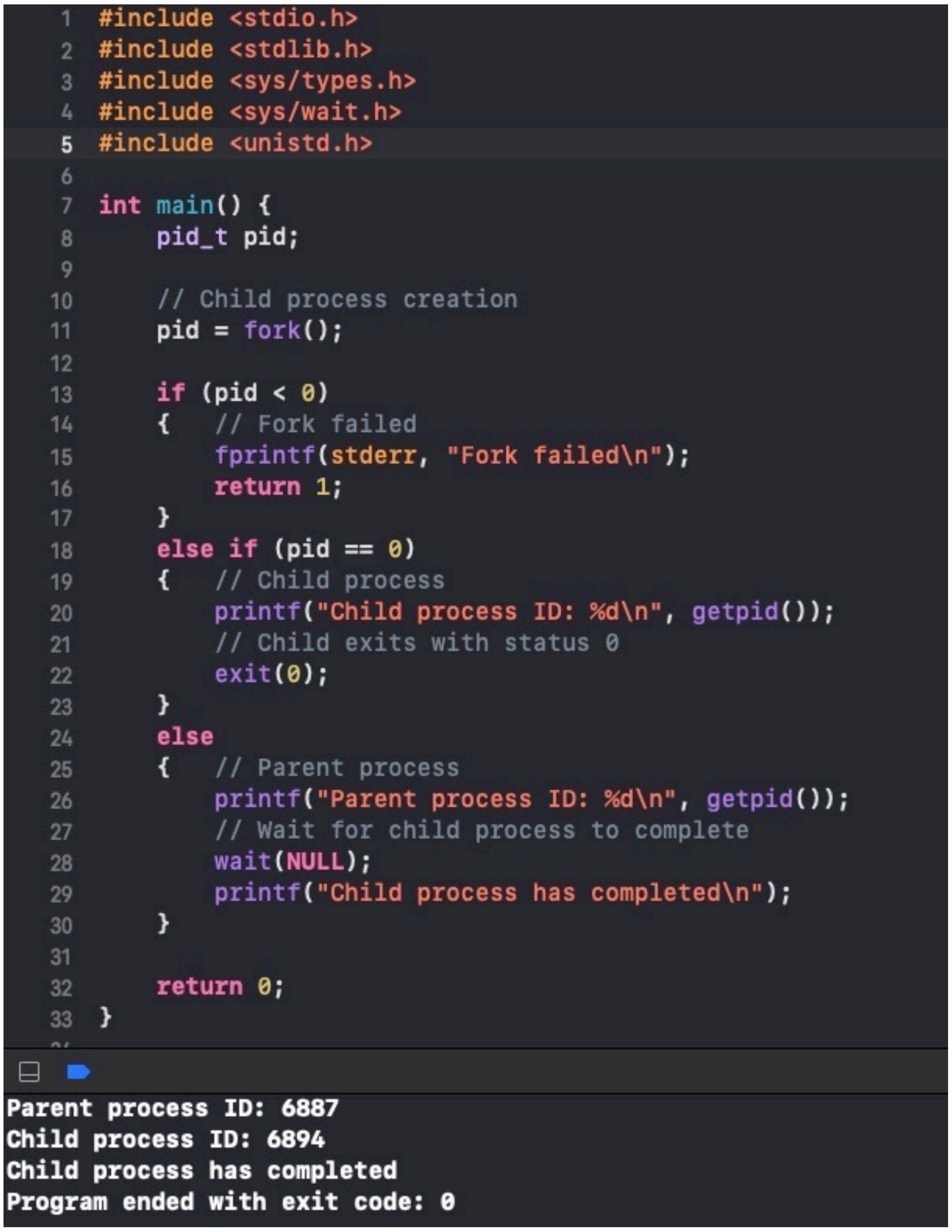


```
[a_1523@eros ~]$ uptime
 23:50:11 up 7 days,  5:26,  9 users,  load average: 0.00, 0.00, 0.00
```

Problem 6

Write a C program that uses fork() to create a child process and have the parent process waits for the child process to run until completion (using wait()). The child would simply print its process id (using getpid()) and returns a value 0 (using exit()). The parent process would then print its own process ID and exits as well.[**4 pts**]

For this problem, you are allowed to lookup the parameters and return values for these system calls.



```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
    pid_t pid;

    // Child process creation
    pid = fork();

    if (pid < 0)
    {   // Fork failed
        fprintf(stderr, "Fork failed\n");
        return 1;
    }
    else if (pid == 0)
    {   // Child process
        printf("Child process ID: %d\n", getpid());
        // Child exits with status 0
        exit(0);
    }
    else
    {   // Parent process
        printf("Parent process ID: %d\n", getpid());
        // Wait for child process to complete
        wait(NULL);
        printf("Child process has completed\n");
    }

    return 0;
}
```

```
Parent process ID: 6887
Child process ID: 6894
Child process has completed
Program ended with exit code: 0
```