HW 1 -3600 Operating System

Submit your answers in as a pdf document on or before (02/13/2021).

1. Indicate whether each series of state transitions for a process is valid or invalid. Justify your answer if its invalid. **(3 points)**
    a. new ➛ ready ➛ blocked ➛ ready
    b. running ➛ blocked ➛ ready ➛ blocked
    c. new ➛ ready ➛ running ➛ ready

a. Invalid, can only be set to running or suspended after the ready state, can't jump to blocked.
b. Invalid, Same reason as 'a'
c. Valid, new process can be set to ready, then dispatch to run, then back to ready with a timeout

2. From the list, indicate which PCB fields will not change during a **process's lifetime**. Why? **(2 points)**
Child, Parent, CPU_state and process_state

The parent will always be the same considering it creates the child processes and ends them as well. The CPU state has multiple states it changes to as well, a good example of this would be the IO states it enters for the user.

3. From the list, indicate which PCB fields may change while a process is in the **running state.** Why? **(2 points)**
Child, Parent, CPU_state and process_state

The Child and Process State. As mentioned earlier, the child is created and deleted often by the parent, this case would be created as its running. The process_state is literally the state of the processes whether the child or parent, thus causing it to change in the running state once it's not in that state anymore or changing from ready to running initially.

4. Learn the **top** command to display the resource utilization statistics of processes
    - Open a terminal and type the **top** command.
    - Start a browser and see the effect on the top display.
    - Press Ctrl-Z to stop.

4.1 Observe and Write what did you notice? **(2 points)**

Very similar to task manager in Windows. It displays the top process taking up the most cpu power. Opening firefox puts it to the top as nothing is running right before. Other processes open up in the back as well, assuming this is for purely the browser (history, cache, gui...etc).

4.2 Compile any C program (Use a long loop - program2 given below using while (1) loop) to observe the effect) **in a new terminal** and observe the effect in top command in the old terminal. Write your observations **(2 points)**

The process took around 21-23% of the CPU. Assuming that it keeps all the data it's recording, the cpu percentage would rise as time goes on if left on. The terminal "Gnome" is on the top as well. The CPU rises to 35% as well when the program is running. totalling to half of the CPU

4.3 From the top display, answer the following: – How much memory is free in the system? – Observe Which process is taking more CPU? – Which process has got maximum memory share? **(4 points)**

I have currently 6447.1 MiB free. Usually on the top would be "systemd", after research this might be the actual OS. Gnome-t ends up on the top of the list as well as the terminal runs

4.4 Write a CPU bound C program (Program 1) and an I/O bound C program (program2 using more printf statements within while (1) loop), compile and execute both of them. Observe the effect of their CPU share using the top display and comment. **Press Ctrl -Z to stop any running program. (5 points)**

The CPU pops up to 75% of usage. The usage of memory from both programs aren't intense enough to see a significant difference. Both programs end up on the top of the list of the processes.

Program 1

```
#include <unistd.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
  unsigned int i,j;
  while(1)
    {
      j = 1;
      for (i = 1; i <= 10; i++)
        {
            j = j*i;
         }
   }
```

```
}
```

Program 2

```
#include <stdio.h>
#include <sys/time.h>

int main(int argc, char *argv[])
{
  unsigned int i;
  int count = 0;
  struct timeval tv;
    while(1)
    {
      for(i = 0; i < 10; i++)
   {
     gettimeofday(&tv, NULL);
     printf("%lu sec, %lu usec\n", tv.tv_sec, tv.tv_usec);
   }
      count++;
      printf("round %d complete\n", count);
    }
}
```