Homework No.1

CSC 345

1.The issue of resource utilization shows up in different forms in different types of operating systems. List what resources must be managed carefully in the following settings:

(10%)

1. Server systems

* Memory allocation with a focus on parallelism
* CPU scheduling with a focus on parallelism
* Security / User-group management
* Network communications
* File management
* Basic IO
  + Depending on the server, may be CLI or Desktop

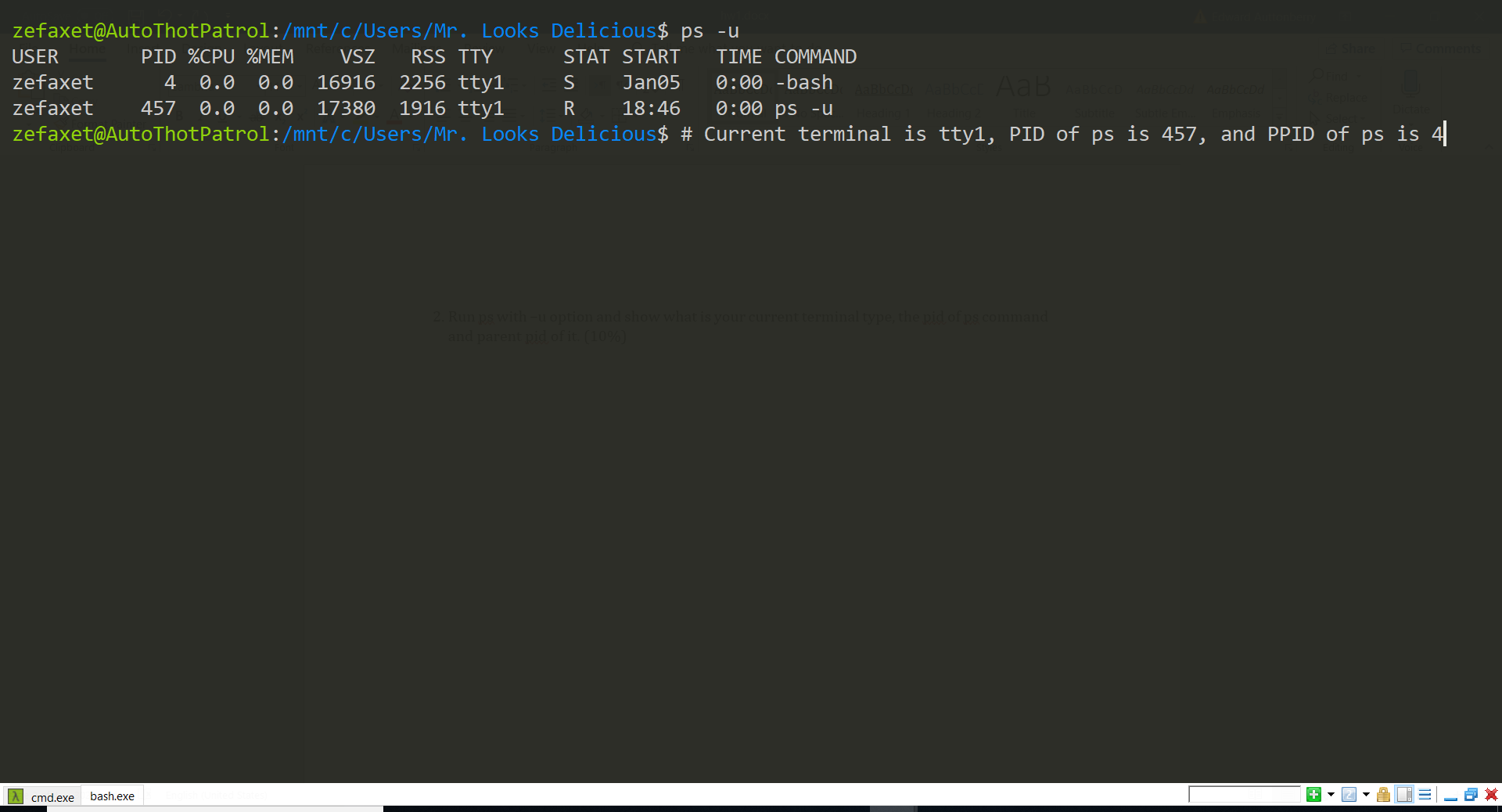
1. Desktops connected to servers

* Network communications
* GUI
* User-group management on a smaller scale
* Process Management with a focus on efficient user applications
* File Management

1. Tablet

* Network communications
* UI (support for touch screen)
* Power Management
* Memory Management

1. Run ps with –u option and show what is your current terminal type, the pid of ps command and parent pid of it. (10%)



1. Enter the following commands and report your observation. (10%)
2. who and tty

who shows information about users logged into the system

tty shows which terminal is currently connected to stdout

1. tput clear

This clears the screen. Seems to be equivalent to $clear.

1. id

Shows user and group ids.

1. ps

shows running processes and their ids

1. echo $$

outputs the PID of the shell

4. List five services provided by an operating system. Explain how each provides convenience to the users. Explain also in which cases it would be impossible for user-level programs to provide these services and how they are protected.(10%)

* CPU Scheduling
  + Keeps multiple processes running efficiently in parallel without the concern of the user.
* Memory Allocation
  + Allocates necessary amounts of memory to specific processes varying with priority and user’s desire.
  + This is not possible in most user-level programs because it would be dangerous to allow user processes to define their own memory space. User level programs are also not able to interface directly with memory modules. They would also not be able to prevent themselves from using another process’s memory.
  + Ensures that processes are limited to their allocation memory space to prevent malicious or accidental invasion of sensitive memory.
* File Management
  + Simplifies the process of mounting storage volumes for higher-level processes to use. Defines how higher-level processes access files.
  + User applications would not be able to mount drives because they can not interface with the hardware
* IO
  + Provides a ground-level framework for various IO methods including Desktop GUI, CLI, and streaming information to and from memory. The system uses this framework to define how user-applications utilize physical memory and display devices.
* Security Roles
  + Provides a method of assigning roles and permissions to specific users or groups of users to ensure efficient and intended system interactions
  + This is not possible for user-level applications because they would not be able to ensure roles are observed by users for other user-level applications. The operating system should act as the primary broker in this regard.

5. Explain mechanisms that OS and hardware provides to ensure system resources and instructions are protected. (10%)

* Dual-Mode operation separates instructions into two modes: user mode and kernel mode. This is performed using a mode bit that is integrated into the hardware. Interrupts from system calls and hardware switches flip the mode bit to 0 to signal that instructions are to run in kernel mode. Certain privileged instructions can only be executed in kernel mode.
* IO Protection works under the pretense that all IO operations run as kernel mode instructions. This protocol simply ensures that the user does not have control of the system while instructions are being executed in kernel mode by locking the user until IO instructions are complete and the mode bit returns to 1.
* Memory protection utilizes two registers to define the program space allocated to a user application. One register defines the starting address usable by a user application and the other defines the range of the allocation. User application are restricted to this memory space. These registers can only be using privileged instructions run in kernel mode.
* CPU protection defines a protocol wherein an internal digital timer is used to interrupt running processes to ensure that the operating system maintains control. An interrupt signal is sent to a given CPU if that CPU’s timer decrements to zero. Setting the timer can only be done in kernel mode.

6. Describe the actions a kernel takes to context switch between processes. (10%)

Given the two PCB’s reserved for the states for the two respective processes, the state of the process being switched from is saved into the PCB reserved for it. A state is then read from the other PCB which is used to begin/continue execution of the new process.

7. (40%, 10% each)

Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

|  |  |  |
| --- | --- | --- |
| Process | Burst Time | Priority |
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0.

1. Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FCFS | | | | | |
| P1 | | P2 | P3 | P4 | P5 |
| 0 | 10 | 11 | 13 | 14 | 19 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SJF | | | | | |
| P4 | P3 | | P1 | P5 | P2 |
| 0 | 1 | 3 | 13 | 18 | 19 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nonpreemptive | | | | | |
| P2 | P5 | | P3 | P1 | P4 |
| 0 | 1 | 6 | 8 | 18 | 19 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RR (Quantum = 1) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| P1 | | P2 | P3 | | P4 | | P5 | | P1 | | P3 | | P5 | | P1 | | P5 | | P1 | | P5 | | P1 | | P5 | | P1 | | P1 | | P1 | | P1 | | P1 | |
| 0 | 1 | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | |

1. What is the turnaround time of each process for each of the scheduling algorithms in part a?

* FCFS:
  + P1: 10
  + P2: 11
  + P3: 13
  + P4: 14
  + P5: 19
* SJF:
  + P1: 13
  + P2: 19
  + P3: 3
  + P4: 1
  + P5: 18
* Nonpreemptive:
  + P1: 18
  + P2: 1
  + P3: 8
  + P4: 19
  + P5: 6
* RR:
  + P1: 19
  + P2: 2
  + P3: 7
  + P4: 4
  + P5: 14

1. What is the waiting time of each process for each of the scheduling algorithms in part a?

* FCFS: Avg 9.6
  + P1: 0
  + P2: 10
  + P3: 11
  + P4: 13
  + P5: 14
* SJF: Avg 7
  + P1: 3
  + P2: 18
  + P3: 1
  + P4: 0
  + P5: 13
* Nonpreemptive: Avg 6.6
  + P1: 8
  + P2: 0
  + P3: 6
  + P4: 18
  + P5: 1
* RR: Avg 5.4
  + P1: 9
  + P2: 1
  + P3: 5
  + P4: 3
  + P5: 9

1. Which of the schedules in part a results in the minimal average waiting time (over all processes)?

RR