**Concepts of Operating System**

**Assignment 2**

**What will the following commands do?**

**echo "Hello, World!":**

**The echo command is a built-in feature in**[**Linux**](https://phoenixnap.com/kb/what-is-linux)**that prints out its arguments as standard output. It is used to display text strings or the command results.**

**name="Productive"**

**touch file.txt:**

**It is used to create a file without any content. The file created using touch command is empty. This command can be used when the user doesn’t have data to store at the time of file creation.**

**ls -a:**

**The "ls" command in Linux, which stands for 'list', lists any normal files within the current directory.**

**'ls -a', it will list any files including hidden files in the current directory.**

**rm file.txt:**

**rm file.txt Running this command removed the file file.txt.**

**cp file1.txt file2.txt:**

**This command will copy both file1.txt and file2.txt into the specified destination directory.**

**mv file.txt /path/to/directory/:**

**To move a file to a different directory, all you have to do is specify the filename or the path to the file and then the path to the destination directory**

**chmod 755 script.sh:**

**Only owner can write, read and execute for everyone**

**grep "pattern" file.txt:**

**It will display all lines with your found.** **If you just need some specific informations of that line.**

**kill PID:**

**Here is the command to kill the Java process by is Process Name instead of its ProcessID.**

**11.**

**12.** **ls -l | grep ".txt":**

**grep is a command-line utility that searches for a specific text string in one or more files.**

**13.** **cat file1.txt file2.txt | sort | uniq:**

**14.** ** ls -l | grep "^d":**

**it lists only directories in the current directory.**

**15.** **grep -r "pattern" /path/to/directory/:**

**If you want to search for a specific pattern from every file present in the directory, then you can use the grep command recursively, and it will search the given pattern from every file present in that specific directory.**

**16.** **cat file1.txt file2.txt | sort | uniq –d:**

**17:** **chmod 644 file.txt:**

**chmod 644’ sets the permissions of a file in Unix so that the owner can read and write, while the group and others can only read, used with the syntax chmod 644 .**

**18.** **cp -r source\_directory destination\_directory:**

**This command copies the entire source\_directory and all its subdirectories and files to the specified destination directory.**

**19.** **find /path/to/search -name "\*.txt":**

**20.** **chmod u+x file.txt:**

**To make a file executable in Linux, the executable mode bit needs to be enabled. To set the executable mode bit, the chmod command is used**

**21.** **echo $PATH:**

**View your PATH**

**Sometimes, you may wish to install programs into other locations on your computer, but be able to execute them easily without specifying their exact location. You can do this easily by adding a directory to your $PATH.**

**Part B**

**Identify True or False:**

**ls is used to list files and directories in a directory.**

**True**

**mv is used to move files and directories.**

**True**

**cd is used to copy files and directories.**

**False**

**pwd stands for "print working directory" and displays the current directory.**

**True**

**5. grep is used to search for patterns in files.**

**True**

**6.chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.**

**True**

**7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.**

**True**

**8. rm -rf file.txt deletes a file forcefully without confirmation.**

**True**

**Identify the Incorrect Commands:**

**chmodx is used to change file permissions.**

**cpy is used to copy files and directories.**

**mkfile is used to create a new file.**

**catx is used to concatenate files.**

**Incorrect Command**

**5.rn is used to rename files.**

**Incorrect Command**

**Part C**

**Question 1: Write a shell script that prints "Hello, World!" to the terminal.**

**Ans:** **$ nano HelloWorld.sh**

**#!/bin/sh**

**echo " Hello World "**

**$ ./Helloworld.sh**

**Hello World**

**Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.**

**Ans:** **name=” CDAC Mumbai "**

**echo "$name"**

**Question 3: Write a shell script that takes a number as input from the user and prints it.**

**Ans:** **#!/usr/bin/env bash**

**read name**

**echo "Hello, $name"**

**./demo.sh**

**Remo**

**Hello ,Remo**

**Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.**

**Ans:** **#!/bin/bash**

**a=5**

**b=3**

**sum=$(( $a + $b ))**

**echo "Sum is: $sum"**

**Output: Sum is:8**

**Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise**

**prints "Odd".**

**Ans:** **echo "Enter a number:"**

**read n**

**echo "RESULT: "**

**if [ `expr $n % 2` == 0 ]**

**then**

**echo "$n is even"**

**else**

**echo "$n is Odd"**

**fi**

**Output: Enter a number:23**

**RESULT: 23 is Odd**

**Question 6: Write a shell script that uses a for loop to print numbers from 1 to5**

**Ans:**

**for i in 1 2 3 4 5;**

**do**

**echo $i;**

**done**

**Output:**

**1**

**2**

**3**

**4**

**5**

**Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5**

**Ans: *#!/usr/bin/bash***

***i=1***

***# iterate the loop***

***while [ $i -le 5 ];  
do  
echo $i  
((i++))  
done***

**Output:**

**Bash/while.sh**

**1**

**2**

**3**

**4**

**5**

**Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".**

**Ans:**

**#!/bin/bash**

**if [ -f "File.txt" ];**

**then**

**echo "File is exist"**

**else**

**echo "File is not exist"**

**fi**

**Output:**

**$ chmod +x ./FirstFile.sh**

**$ ./FirstFile.sh**

**File is exist**

**Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.**

**Ans:**

**echo "enter numbers";**

**read a ;**

**echo "a=$a";**

**if [ $a \> $10 ];**

**then**

**echo "a is greater than 10";**

**else**

**echo "10 is greater than a";**

**fi;**

**Output: $ ./FirstFile.sh**

**a=11**

**11 is greater than 10**

**Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.**

**Ans:**

**#!/ in/bash**

**for i in 1 2 3 4 5**

**do**

**for j in 1 2 3 4 5**

**do**

**if [ $j -le $i ]**

**then**

**echo -ne "$i"X"$j"=`expr "$i"\*"$j"` "\t"**

**else**

**echo**

**break**

**fi**

**done**

**done**

**echo -ne "\n"**

**output:**

**1X1=1\*1**

**2X1=2\*1 2X2=2\*2**

**3X1=3\*1 3X2=3\*2 3X3=3\*3**

**4X1=4\*1 4X2=4\*2 4X3=4\*3 4X4=4\*4**

**5X1=5\*1 5X2=5\*2 5X3=5\*3 5X4=5\*4 5X5=5\*5**

**Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.**

**Ans:#!/bin/bash**

**positive\_number() {**

**local number="$1"**

**if [ "$number" -gt 0 ]; then**

**echo "Valid positive number. Exiting the loop."**

**return 0**

**else**

**echo "Invalid number. Please enter a positive number."**

**return 1**

**fi**

**}**

**while true; do**

**read -p "Enter a positive number: " n**

**if positive\_number "$n"; then**

**break**

**fi**

**done**

**output:./file1.txt**

**Enter a positive number:-1**

**Invalid number. Please enter a positive number**

**Enter a positive number:2**

**Valid positive number. Exiting the loop**

**Part D**

**What is an operating system, and what are its primary functions**

**Ans:** **Hardware Manager: It manage all the hardware resources or components of computer.**

**Process Manager: It supervise all the task/process/job which is being executed by processor.**

**Functions of OS:**

**Process Management (Process Scheduling Algo)**

**Memory Management**

**Device Management (HDD, Printer, Monitor, Spealer, WebCam)**

**Disk Management (Disk Scheduling Algos)**

**Network Management (Network Card / Controller)**

**Security Management (Firewall, Anti Virus, Anti Spyware)**

**File Management**

**3.What is virtual memory, and how does it work?**

**Ans:** **Its a memory space in hard-drive, which work like physical memory to entertain large processes whose size is bigger than the RAM. It is an illusion memory**

**In Virtual Memory process is divided into fixed sized partions know as pages.**

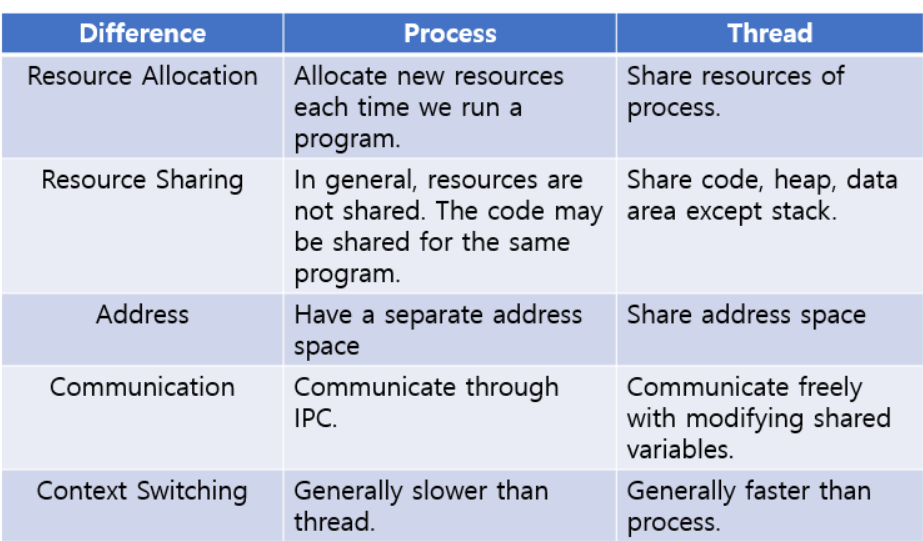
**OS loads the process's pages from virtual memory to physical memory on demand of CPU i.e. known as Demand Paging.**

**While working with virtual memory pages are saved as per their logical address. While loading the pages from virtual memory to physical memory the pages get physical address.**

**If a page loaded from virtual memory to physical memory i.e. know as swap-in process**

**3.Explain the difference between process and thread.**

**Ans:**



**4.Describe the difference between multiprogramming, multitasking, and multiprocessing.**

**Ans:**

|  |  |  |
| --- | --- | --- |
| **Multiprogramming** | **Multitasking** | **Multiprocessing** |
| **Running multiple programs on a single CPU** | **Running multiple tasks (applications) on a single CPU)** | **Running multiple processes on multiple CPUs (or cores)** |
| **Resources (CPU, memory) are shared among programs** | **Resources (CPU, memory) are shared among tasks** | **Each process has its own set of resources (CPU, memory)** |
| **Each program has its own memory space** | **Each task has its own memory space** | **Each process has its own memory space** |

**5.What is a file system, and what are its components?**

**Ans: A file system is a way of organizing and managing files on a storage device such as a hard disk or a flash drive. It provides a logical structure to the physical storage space and allows users and applications to access and manipulate the files.**

**A file system typically consists of three components:**

**files, directories, and file metadata. The file system hierarchy starts with a root directory and includes subdirectories that can contain files and further subdirectories.**

**6.What is a deadlock, and how can it be prevented?**

**Ans:** **A deadlock is a situation where a set of processes is blocked because each process is holding a resource and waiting for another resource acquired by some other process.**

**how can it be prevented:**

**1.Eliminate Mutual Exclusion**

**2. Eliminate Hold and Wait**

**3. Eliminate No Preemption**

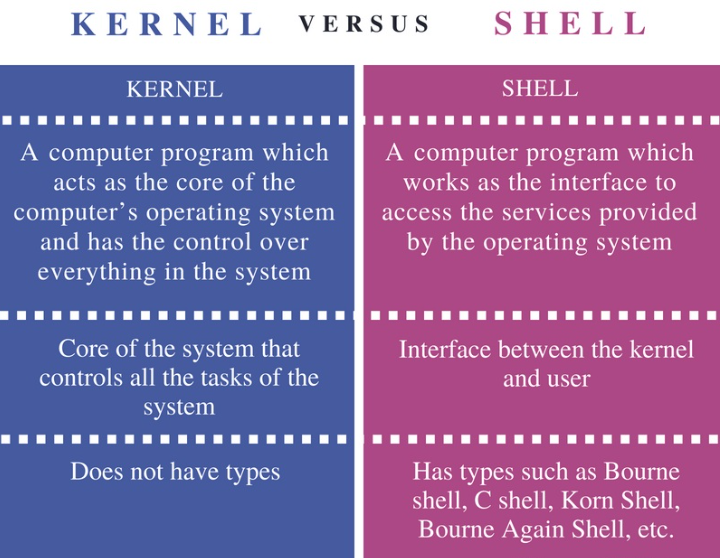
**Eliminate Circular Wait**

**Detection and Recovery**

**What is CPU scheduling, and why is it important?**

**Ans: CPU Scheduling is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I / O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer.**

**8.Explain the difference between a kernel and a shell.**



**9.How does a system call work?**

**Ans: Users need special resources: Sometimes programs need to do some special things that can’t be done without the permission of the OS like reading from a file, writing to a file, getting any information from the hardware, or requesting a space in memory.**

**The program makes a system call request: There are special predefined instructions to make a request to the operating system. These instructions are nothing but just a “system call”. The program uses these system calls in its code when needed.**

**Operating system sees the system call: When the OS sees the system call then it recognizes that the program needs help at this time so it temporarily stops the program execution and gives all the control to a special part of itself called ‘Kernel’. Now ‘Kernel’ solves the need of the program.**

**The operating system performs the operations: Now the operating system performs the operation that is requested by the program. Example: reading content from a file etc.**

**Operating system give control back to the program : After performing the special operation, OS give control back to the program for further execution of program .**

**10.** **What is the purpose of device drivers in an operating system**

**Ans: Device Drivers are essential for a computer system to work properly because without a device driver the particular hardware fails to work accordingly, which means it fails in doing the function/action it was created to do. Most use the term Driver, but some may say Hardware Driver, which also refers to the Device Driver.**

**11.Explain the role of the page table in virtual memory management.**

**Ans:** **Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.**

**12.What is thrashing, and how can it be avoided**

**Ans: Thrashing is a condition or a situation when the system is spending a major portion of its time servicing the page faults, but the actual processing done is very negligible.**

**Thrashing is a phenomenon that occurs in computer systems, particularly in virtual memory systems, when the operating system spends a significant amount of time swapping data between physical memory (RAM) and disk storage.**

**how can it be avoided: Increase Physical Memory, Optimize Memory Usage,**

**Use Working Set Algorithms, Adjust Process Priorities**

**13.**

**Increase Physical Memory**

**Part E**

**Consider the following processes with arrival times and burst times:**

**| Process | Arrival Time | Burst Time |**

**|---------|--------------|------------|**

**| P1 | 0 | 5 |**

**| P2 | 1 | 3 |**

**| P3 | 2 | 6 |**

**Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.**

**Ans: average waiting time:3.3**

**Consider the following processes with arrival times and burst times:**

**| Process | Arrival Time | Burst Time |**

**|---------|--------------|------------|**

**| P1 | 0 | 3 |**

**| P2 | 1 | 5 |**

**| P3 | 2 | 1 |**

**| P4 | 3 | 4 |**

**Calculate the average turnaround time using Shortest Job First (SJF) scheduling.**

**Ans: average turnaround time :5.5**

**Consider the following processes with arrival times, burst times, and priorities (lower number**

**indicates higher priority):**

**| Process | Arrival Time | Burst Time | Priority |**

**|---------|--------------|------------|----------|**

**| P1 | 0 | 6 | 3 |**

**| P2 | 1 | 4 | 1 |**

**| P3 | 2 | 7 | 4 |**

**| P4 | 3 | 2 | 2 |**

**Calculate the average waiting time using Priority Scheduling.**

**Ans: average waiting time:5**

**Consider the following processes with arrival times and burst times, and the time quantum for**

**Round Robin scheduling is 2 units:**

**| Process | Arrival Time | Burst Time |**

**|---------|--------------|------------|**

**| P1 | 0 | 4 |**

**| P2 | 1 | 5 |**

**| P3 | 2 | 2 |**

**| P4 | 3 | 3 |**

**Calculate the** **average turnaround time using Round Robin scheduling.**

**average turnaround time:5.56**

**5.Consider a program that uses the fork() system call to create a child process. Initially, the parent process has a variable x with a value of 5. After forking, both the parent and child processes**

**#include <stdio.h>**

**#include <unistd.h>**

**#define SIZE 5**

**int nums[SIZE] = {0,1,2,3,4};**

**int main()**

**{**

**int i;**

**pid\_t pid;**

**pid = fork();**

**if (pid == 0) {**

**for (i = 0; i < SIZE; i++) {**

**nums[i] \*= -i;**

**printf("CHILD: %d ",nums[i]);**

**}**

**}**

**else if (pid > 0) {**

**wait(NULL);**

**for (i = 0; i < SIZE; i++)**

**printf("PARENT: %d ",nums[i**

**}**

**return 0;**

**}**