Q1.Explore usage of basic Linux Commands for files, directory and process management

Sure! Here are some basic Linux commands for file, directory, and process management:

1. ls: Lists the files and directories in the current directory.

Example: `ls`

2. cd: Changes the current directory.

Example: `cd /home/user/Documents`

3. pwd: Prints the current working directory.

Example: `pwd`

4. mkdir: Creates a new directory.

Example: `mkdir new_directory`

5. rmdir: Removes an empty directory.

Example: `rmdir directory_to_remove`

6. rm: Removes a file or directory (with the -r flag).

Example: `rm file_to_remove` or `rm -r directory_to_remove`

7. cp: Copies a file or directory.

Example: `cp file_to_copy new_file` or `cp -r directory_to_copy new_directory`

8. mv: Moves a file or directory.

Example: `mv file_to_move new_location` or `mv directory_to_move new_location`

9. cat: Displays the contents of a file.

Example: `cat file_to_display`

10. less: Displays the contents of a file one page at a time.

Example: `less file_to_display`

11. ps: Lists the current running processes.

Example: `ps`

12. kill: Terminates a process.

Example: `kill process_id`

These are just a few of the basic Linux commands for file, directory, and process management. There are many more commands available, and each one has its own specific usage and options.

Q2. Linux Shell Script

· Display OS version, release number, kernel version

· Display top 10 processes in descending order

· Display processes with the highest memory usage

Ans:

```
#!/bin/sh
echo "--------------------"
echo "Shell Script OSL Practical 2"
echo "OS INFO: "
cat /etc/os-release
uname -r
uname -a
uname -v
echo "--------------------------------"
echo "10 Processes in descending order: "
ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -10
echo "--------------------------------"
echo "Highest Memory Usage: "
ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -2
echo "--------------------------------"
echo "User Login and Name: "
whoami
logname
echo "$SHELL"
pwd
uname -srm
echo "--------------------------------"
```

Q3. Implement linux “cp” command to copy one file to another using kernel APIs

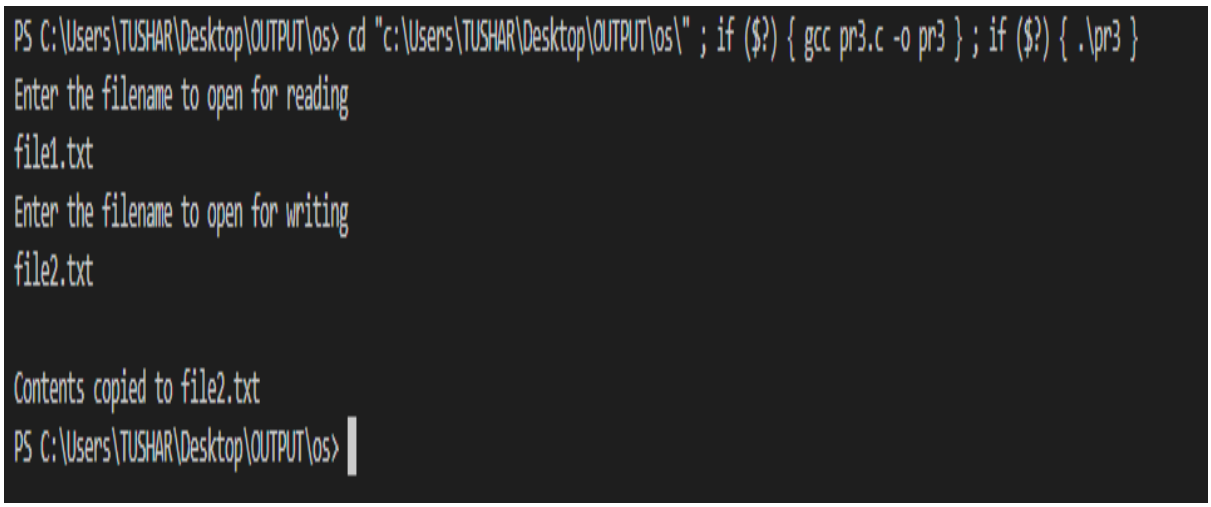
```
#include <stdio.h>
#include <stdlib.h>

int main() {
   FILE *fptr1, *fptr2;
   char filename[100], c;

   printf("Enter the filename to open for reading\n");
   scanf("%s", filename);

   // Open the input file for reading
   fptr1 = fopen(filename, "r");
   if (fptr1 == NULL) {
      printf("Cannot open file %s\n", filename);
      exit(0);
   }

   printf("Enter the filename to open for writing\n");
   scanf("%s", filename);

   // Open the output file for writing
   fptr2 = fopen(filename, "w");
   if (fptr2 == NULL) {
      printf("Cannot open file %s\n", filename);
      exit(0);
   }
```

```
    // Copy the contents of the input file to the output file
    c = fgetc(fptr1);
    while (c != EOF) {
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }

    printf("\nContents copied to %s", filename);

    // Close the files
    fclose(fptr1);
    fclose(fptr2);

    return 0;
}
```

To run :

gcc program_name.c

./a.out



Q4. Create a child process using fork system call. Obtain process ID of both child and parent using getpid and getppid system call.

‘

```
#include <stdio.h>
#include <stdlib.h>
int fork();
int getpid();
int getppid();
int main()
{
   int pid=fork();
   if(pid==0)
   {
   printf("THIS IS THE CHILD PROCESSS.MY PID IS %d and my parent id is:%d\n",getpid(),getppid());
   }
   else
   {
      printf("this is the parent process .my pid is %d and my parent is %d.\n",getpid(),pid);
   }
   return 0;
}
```

Q5. Process management: Scheduling · Write a program to demonstrate the concept of pre-emptive scheduling algorithms (Round Robin Algorithm)

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

void main()
{
    // initlialize the variable name
    int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10],
temp[10];
    float avg_wt, avg_tat;
    printf(" Total number of process in the system: ");
    scanf("%d", &NOP);
    y = NOP; // Assign the number of process to variable y

// Use for loop to enter the details of the process like Arrival time and the
Burst Time
for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");  // Accept arrival time
scanf("%d", &at[i]);
printf(" \nBurst time is: \t"); // Accept the Burst time
scanf("%d", &bt[i]);
temp[i] = bt[i]; // store the burst time in temp array
}
// Accept the Time qunat
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
// Display the process No, burst time, Turn Around Time and the waiting time
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0) // define the conditions
{
    sum = sum + temp[i];
    temp[i] = 0;
    count=1;
    }
    else if(temp[i] > 0)
    {
        temp[i] = temp[i] - quant;
        sum = sum + quant;
    }
    if(temp[i]==0 && count==1)
    {
        y--; //decrement the process no.
```

```
        printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i],
sum-at[i], sum-at[i]-bt[i]);
        wt = wt+sum-at[i]-bt[i];
        tat = tat+sum-at[i];
        count =0;
    }
    if(i==NOP-1)
    {
        i=0;
    }
    else if(at[i+1]<=sum)
    {
        i++;
    }
    else
    {
        i=0;
    }
}
// represents the average waiting time and Turn Around time
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
getch();
}
```

Output:

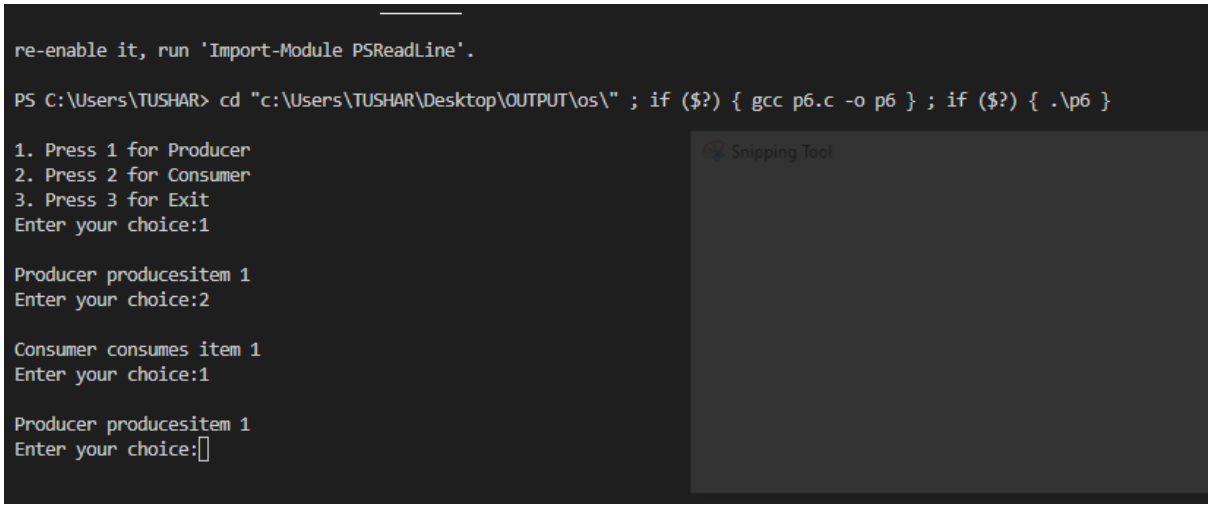
```
C:\Users\TUSHAR\AppData\Local\Temp\cc6ToHup.o:p4.c:(.text+0x10): undefined reference to `fork'
C:\Users\TUSHAR\AppData\Local\Temp\cc6ToHup.o:p4.c:(.text+0x20): undefined reference to `getppid'
collect2.exe: error: ld returned 1 exit status
PS C:\Users\TUSHAR\Desktop\OUTPUT\os> cd "c:\Users\TUSHAR\Desktop\OUTPUT\os\" ; if ($?) { gcc p5.c -o p5 } ; if ($?) { .\p5 }
 Total number of process in the system: 5

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       2

Burst time is:  2

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       3

Burst time is:  1

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       4

Burst time is:  5

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       1

Burst time is:  2

 Enter the Arrival and Burst time of the Process[5]
 Arrival time is:       5

Burst time is:  3
Enter the Time Quantum for the process:         3

 Process No             Burst Time             TAT            Waiting Time
Process No[1]           2                              0                       -2
```

Q6. Write a C program to implement a solution to the Producer consumer problem through Semaphore.

```
 #include <stdio.h>

#include <stdlib.h>
int mutex = 1;
int full = 0;
int empty = 10, x = 0;
void producer()
{
    --mutex;
    ++full;
    --empty;
    x++;
    printf("\nProducer produces"
        "item %d",x);
    ++mutex;
}
void consumer()
{
    --mutex;
    --full;

    ++empty;
    printf("\nConsumer consumes ""item %d",x);
    x--;
    ++mutex;
}
int main()
{
    int n, i;
    printf("\n1. Press 1 for Producer"
        "\n2. Press 2 for Consumer"
        "\n3. Press 3 for Exit");
    for (i = 1; i > 0; i++) {
        printf("\nEnter your choice:");
        scanf("%d", &n);
        switch (n) {
        case 1:
            if ((mutex == 1)
                && (empty != 0)) {
                producer();
            }
            else {
                printf("Buffer is full!");
            }
            break;
```

```
        case 2:
            if ((mutex == 1)
                && (full != 0)) {
                consumer();
            }
            else {
                printf("Buffer is empty!");
            }
            break;
        case 3:
            exit(0);
            break;
        }
    }
}
```

Output:



Q7. Write a program to demonstrate the concept of deadlock avoidance through Banker's Algorithm.

```
#include <stdio.h>
int main()
{
    int n, m, i, j, k;
    n = 5;
    m = 3;
    int alloc[5][3] = { { 0, 1, 0 },
            { 2, 0, 0 },
            { 3, 0, 2 },
            { 2, 1, 1 },
            { 0, 0, 2 } };

    int max[5][3] = { { 7, 5, 3 },
                    { 3, 2, 2 },
                    { 9, 0, 2 },
                    { 2, 2, 2 },
                    { 4, 3, 3 } };

    int avail[3] = { 3, 3, 2 };

    int f[n], ans[n], ind = 0;
    for (k = 0; k < n; k++) {
        f[k] = 0;
    }
    int need[n][m];
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    }
    int y = 0;
    for (k = 0; k < 5; k++) {
        for (i = 0; i < n; i++) {
            if (f[i] == 0) {

                int flag = 0;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]){
                        flag = 1;
                        break;
                    }
                }

                if (flag == 0) {
                    ans[ind++] = i;
                    for (y = 0; y < m; y++)
                        avail[y] += alloc[i][y];
```

```
                    f[i] = 1;
                }
            }
        }
    }

    int flag = 1;

    for(int i=0;i<n;i++)
    {
    if(f[i]==0)
    {
        flag=0;
        printf("The following system is not safe");
        break;
    }
    }

    if(flag==1)
    {
    printf("Following is the SAFE Sequence\n");
    for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
    printf(" P%d", ans[n - 1]);
    }


    return (0);
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE  COMMENTS

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to
re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\TUSHAR> cd "c:\Users\TUSHAR\Desktop\OUTPUT\os\" ; if ($?) { g++ pr7.cpp -o pr7 } ; if ($?) { .\pr7 }
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
PS C:\Users\TUSHAR\Desktop\OUTPUT\os> cd "c:\Users\TUSHAR\Desktop\OUTPUT\os\" ; if ($?) { g++ pr7.cpp -o pr7 } ; if ($?) { .\pr7 }
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
PS C:\Users\TUSHAR\Desktop\OUTPUT\os>
```

Q8. Write a program to demonstrate the concept of dynamic partitioning placement algorithms – Best Fit Algorithm

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
if(lowest>temp)
{
ff[i]=j;

lowest=temp;
}
}
}
frag[i]=lowest;
bf[ff[i]]=1;
lowest=10000;
}
printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
for(i=1;i<=nf && ff[i]!=0;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
```

```
getch();
}
```

Output:

```
PS C:\Users\TUSHAR\Desktop\OUTPUT\os> cd "c:\Users\TUSHAR\Desktop\OUTPUT\os\" ; if ($?) { gcc pr8.c -o pr8 } ; if ($?) { .\pr8 }

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File No File Size       Block No        Block Size      Fragment
1               1               2               2               1
2               4               1               5               1
```

Q9. Write a program in C demonstrate the concept of page replacement policies for handling page faults eg: LRU

```
#include <stdio.h>

#define MAX_FRAMES 3 // maximum number of frames in physical memory
#define MAX_PAGES 10 // maximum number of pages in virtual memory

int page_faults = 0; // count of page faults
int page_frames[MAX_FRAMES]; // array to store the page frames
int frame_times[MAX_FRAMES]; // array to store the last access time of each frame

// function to initialize the page frames and frame_times arrays
void init_frames() {
   int i;
   for (i = 0; i < MAX_FRAMES; i++) {
      page_frames[i] = -1; // set all page frames to -1 (empty)
      frame_times[i] = -1; // set all frame access times to -1 (unused)
   }
}

// function to check if a page is present in a frame
int in_frames(int page) {
   int i;
   for (i = 0; i < MAX_FRAMES; i++) {
      if (page_frames[i] == page) {
         return i; // return the index of the page frame
      }
   }
   return -1; // return -1 if page is not in any frame
}
```

```
// function to find the least recently used frame
int lru_frame() {
   int i, min_time = frame_times[0], min_index = 0;
   for (i = 1; i < MAX_FRAMES; i++) {
      if (frame_times[i] < min_time) {
         min_time = frame_times[i];
         min_index = i;
      }
   }
   return min_index; // return the index of the least recently used frame
}

int main() {
   int page_sequence[MAX_PAGES] = {1, 2, 3, 4, 5, 6, 1, 2, 3, 4}; // the page reference
string
   int num_pages = 10; // the number of pages in the reference string
   int i, j, k, page, frame_index, lru_index;

   init_frames(); // initialize the page frames and frame_times arrays

   // loop over the page reference string
   for (i = 0; i < num_pages; i++) {
      page = page_sequence[i];

      // check if the page is already in a frame
      frame_index = in_frames(page);
      if (frame_index != -1) {
         frame_times[frame_index] = i; // update the access time of the frame
      }
```

```
        else {
            // page fault - find the least recently used frame and replace it
            lru_index = lru_frame();
            page_frames[lru_index] = page;
            frame_times[lru_index] = i;
            page_faults++;

            // print the current state of the page frames array
            printf("Page fault: ");
            for (j = 0; j < MAX_FRAMES; j++) {
                if (page_frames[j] == -1) {
                    printf("_ ");
                }
                else {
                    printf("%d ", page_frames[j]);
                }
            }
            printf("\n");
        }
    }

    printf("Total page faults: %d\n", page_faults);

    return 0;
}
```

Output:

```
PS C:\Users\TUSHAR\Desktop\OUTPUT\os> cd "c:\Users\TUSHAR\Desktop\OUTPUT\os\" ; if ($?) { gcc pr9.c -o pr9 } ; if ($?) { .\pr9 }
Page fault: 1 _ _
Page fault: 1 2 _
Page fault: 1 2 3
Page fault: 4 2 3
Page fault: 4 5 3
Page fault: 4 5 6
Page fault: 1 5 6
Page fault: 1 2 6
Page fault: 1 2 3
Page fault: 4 2 3
Total page faults: 10
PS C:\Users\TUSHAR\Desktop\OUTPUT\os>
```