OSL(CSL403)

AY 2021-22

OSL Practical exam questions.

Each student will get one group of questions.

Gr 1

· Write commands to create a file and display the number of lines as output. ·

Command : 1 cat > filename (enter content of file exit by ctrl+c)

write wc -l filename

Write a program to implement FIFO page replacement policies

Ans:-

```
#include<stdio.h>
#include<conio.h>
main()
{
int i,j,k,f, pf=0, count=0,rs[25],m[10],n;

printf("\nEnter the length of reference string--");
scanf("%d",&n);
printf("\nEnter the reference string--");
for(i=0;i<n;i++)
scanf("%d",&rs[i]);
printf("\nEnterno. offrames-- ");
scanf("%d",&f);
for(i=0;i<f;i++)
m[i]=-1;
printf("\nThePage Replacement Process is--\n");
for(i=0;i<n;i++)
{
```

```
for(k=0;k<f;k++)
{
}
if(k==f)
{
}
if(m[k]==rs[i])
break;
m[count++]=rs[i];
pf++;
for(j=0;j<f;j++)
printf("\t%d",m[j]);
if(k==f)
printf("\tPF No. %d",pf);
printf("\n");
if(count==f)
count=0;
}
printf("\nThenumber ofPageFaultsusingFIFOare %d",pf);
getch();
}
```

Gr 2

· Write any five linux commands.

1) man
2) man man —----------rm
3) cal —--------------------cd
4) date—--------------------rmdir
5) mkdir—----------------wc

· Write a program to display first N even numbers (using do-while loop). shell script

```
#Print up to nth number of even series in shell script
clear
echo "------EVEN SERIES------"
echo -n "Enter a number: "
checker=0
read num
while test $checker -le $num
do
```

```
ii=`expr $checker % 2`
      if test $ii -eq 0
      then
                  echo "$checker"
      fi
checker=`expr $checker + 1`

   done
```

Gr 3

· Write five file Manipulation commands.

1) ls , ls-l , ls-a
2) cat file1
3) cp file1 file2
4) mv file1 file2
5) more file1

· Create a child process in Linux using the fork system call. From the child process obtain the process ID of both child and parent by using getpid and getppid system call

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

int main(void) {
 pid_t pid = fork();

 if(pid == 0) {
  printf("Child => PPID: %d PID: %d\n", getppid(), getpid());
  exit(EXIT_SUCCESS);
 }
 else if(pid > 0) {
  printf("Parent => PID: %d\n", getpid());
  printf("Waiting for child process to finish.\n");
  wait(NULL);
  printf("Child process finished.\n");
 }
 else {
  printf("Unable to create child process.\n");
 }
```

```
  return EXIT_SUCCESS;
}
```

Gr 4

· Write a command to search word in file(case sensitive)

Command = grep word file1

Eg grep abc file1

· Write a program to find greatest of three numbers (shell script)

```
#Print the greatest of three numbers
clear
echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3
echo "The Greatest of three Numbers is : "
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
        echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
        echo $num2
else
        echo $num3
fi
```

Gr 5

· Write commands to create a file and display number of characters in the file as output.

Command : create cat > filename.txt/.sh/.c (anything)

Display : wc -m filename.txt/.sh/.c

· Program to arrange numbers in ascending order.

```
#!/bin/bash
echo "Input numbers to sort:"
read -a array
echo ""
let length=${#array[*]}-1
for ((i = 0; i < $length; i++)); do
        for ((j = 0; j < $length - i; j++)); do
```

```
            if [ ${array[j]} -gt ${array[j + 1]} ]; then
            temp=${array[j]}
            array[j]=${array[j + 1]}
            array[j + 1]=$temp
            fi
            done
done
echo "After sort: ${array[*]}"
```

Gr 6

· Write commands to create a file and display number of words in the file as output.

Command : create cat > filename.txt/.sh/.c (anything)

Display : wc -w filename.txt/.sh/.c

· Write a program to Round robin scheduling algorithm

```
#include<stdio.h>
int main()
{
 int count,j,n,time,remain,flag=0,time_quantum;
 int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
 printf("Enter Total Process:\t ");
 scanf("%d",&n);
 remain=n;
 for(count=0;count<n;count++)
 {
  printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
  scanf("%d",&at[count]);
  scanf("%d",&bt[count]);
  rt[count]=bt[count];
 }
 printf("Enter Time Quantum:\t");
 scanf("%d",&time_quantum);
 printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
 for(time=0,count=0;remain!=0;)
 {
  if(rt[count]<=time_quantum && rt[count]>0)
  {
   time+=rt[count];
   rt[count]=0;
   flag=1;
  }
  else if(rt[count]>0)
  {
   rt[count]-=time_quantum;
   time+=time_quantum;
```

```
    }
    if(rt[count]==0 && flag==1)
    {
     remain--;
     printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
     wait_time+=time-at[count]-bt[count];
     turnaround_time+=time-at[count];
     flag=0;
    }
    if(count==n-1)
     count=0;
    else if(at[count+1]<=time)
     count++;
    else
     count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}
```

Gr 7

· Write command to see months in ascending order in the Linux System.

Create cat > months.txt

Add months

sort months.txt

· Write a program to implement dynamic partitioning placement algorithms best fit

```
#include<stdio.h>

int main()
{
    int fragments[10], block[10], file[10], m, n, number_of_blocks, number_of_files, temp,   lowest = 10000;
    static int block_arr[10], file_arr[10];
    printf("\nEnter the Total Number of Blocks:\t");
    scanf("%d", &number_of_blocks);
    printf("\nEnter the Total Number of Files:\t");
    scanf("%d", &number_of_files);
    printf("\nEnter the Size of the Blocks:\n");
    for(m = 0; m < number_of_blocks; m++)
    {
        printf("Block No.[%d]:\t", m + 1);
        scanf("%d", &block[m]);
    }
    printf("Enter the Size of the Files:\n");
    for(m = 0; m < number_of_files; m++)
    {
        printf("File No.[%d]:\t", m + 1);
```

```
            scanf("%d", &file[m]);
        }
        for(m = 0; m < number_of_files; m++)
        {
            for(n = 0; n < number_of_blocks; n++)
            {
                if(block_arr[n] != 1)
                {
                    temp = block[n] - file[m];
                    if(temp >= 0)
                    {
                        if(lowest > temp)
                        {
                            file_arr[m] = n;
                            lowest = temp;
                        }
                    }
                }
                fragments[m] = lowest;
                block_arr[file_arr[m]] = 1;
                lowest = 10000;
            }
        }
        printf("\nFile Number\tFile Size\tBlock Number\tBlock Size\tFragment");
        for(m = 0; m < number_of_files && file_arr[m] != 0; m++)
        {
            printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", m, file[m], file_arr[m], block[file_arr[m]], fragments[m]);
        }
        printf("\n");
        return 0;
    }
```

Gr 8

. Create a file, copy the file in XYZ folder and rename that file with the new name. ·

. Write A Program for FIFO scheduling

```
include<stdio.h>

#include<stdlib.h>

void main()

{

        int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;

        printf("Enter total number of processes(maximum 20): ");
```

```
scanf("%d",&n);
        printf("\nEnter the process burst time: ");
        for(i=0;i<n;i++)
        {
        printf("\nP[%d]:",i+1);
        scanf("%d",&bt[i]);
        }
        wt[0]=0;
        for(i=1;i<n;i++)
        {
        wt[i]=0;
        for(j=0;j<i;j++)
        {
              wt[i]+=bt[j];
        }
        }
printf("\nProcess\t\tBurst Time\tWaiting Time\tTurn Around Time");
        for(i=0;i<n;i++)
        {
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
         avtat+=tat[i];
        printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
        }
        avwt/=i;
        avtat/=i;
        printf("\n\nAverage Waiting Time:%d",avwt);
```

```
        printf("\n\nAverage Turn Around Time:%d",avtat);
}
```

Gr 9

· Write a command to create a file .Find out letter ‘a’ and ‘ A’ in file.

Command = grep -i a filename

· Write a program to implement dynamic partitioning placement algorithms worst fit

```
#include<stdio.h>
#define max 25
int main(){
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - Worst Fit");
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) //if bf[j] is not allocated
{
temp=b[j]-f[i];
if(temp>=0)
if(highest<temp){
ff[i]=j;
highest=temp;
}
}
}
frag[i]=highest;
bf[ff[i]]=1;
highest=0;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\n",i,f[i],ff[i],b[ff[i]],frag[i]);
return 0;
```

```
}
```

Gr10

· Write commands to create a file and display number of line as output.

Same as grp -1·

Write a Program for SJF scheduling.

```
include<stdio.h>
int main(){
        int n,p[20],bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j,pos,temp;
        printf("Enter total number of processes(maximum 20): ");
        scanf("%d",&n);
        printf("\nEnter the process burst time: ");
        for(i=0;i<n;i++)
        {
        printf("\nP[%d]:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
        }
        //Sorting the burst time
  for(i=0;i<n;i++){
        for(j =i+1;j<n;j++)
        {
        if(bt[i] > bt[j]){
        temp = bt[i];
        bt[i] = bt[j];
        bt[j] = temp;
        }
        }
        }
        wt[0]=0;
        for(i=1;i<n;i++){
        wt[i]=0;
        for(j=0;j<i;j++){
                wt[i]+=bt[j];
        }
        }
    printf("\nProcess\t\tBurstTime\tWaiting Time\tTurn Around Time");
        for(i=0;i<n;i++){
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
        avtat+=tat[i];
              printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
        }
        avwt/=n;
        avtat/=n;
        printf("\n\nAverage Waiting Time:%d",avwt);
        printf("\n\nAverage Turn Around Time:%d",avtat);
        return 0;
```

}

Gr11

· Write a command to create a file. Return number of words and lines in file ·

Command = wc -l -w filename

Write a program to Round robin scheduling algorithm

Ans :- Grp 6

Gr 12

· Write a command to create a file .Find out letter ‘IS ’ in file.

Command = grep

· Write a program to implement FIFO page replacement policies.

Ans Grp 1 same

Grp 13

· Write a command to create a file. Return number of words and lines in file ·

Same as grp 11

Write a program to implement LRU page replacement policies.

```
#include<stdio.h>
int findLRU(int time[], int n){
int i, minimum = time[0], pos = 0;
for(i = 1; i < n; ++i){
if(time[i] < minimum){
minimum = time[i];
pos = i;
}
}
return pos;
}
```

```
int main()
{
   int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;
printf("Enter number of frames: ");
scanf("%d", &no_of_frames);
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);
printf("Enter reference string: ");
   for(i = 0; i < no_of_pages; ++i){
            scanf("%d", &pages[i]);
   }

for(i = 0; i < no_of_frames; ++i){
            frames[i] = -1;
   }

   for(i = 0; i < no_of_pages; ++i){
            flag1 = flag2 = 0;

            for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == pages[i]){
            counter++;
            time[j] = counter;
  flag1 = flag2 = 1;
  break;
  }
            }

            if(flag1 == 0){
for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == -1){
            counter++;
            faults++;
            frames[j] = pages[i];
            time[j] = counter;
            flag2 = 1;
            break;
            }
            }
            }

            if(flag2 == 0){
            pos = findLRU(time, no_of_frames);
            counter++;
            faults++;
            frames[pos] = pages[i];
            time[pos] = counter;
            }

            printf("\n");

            for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
            }
```

```
}
printf("\n\nTotal Page Faults = %d", faults);

   return 0;
}
```

Gr 14

· Write a command to create a file .Find out letter ‘a’ and ‘ A’ in file. ·

Same as Grp-9

. Write a program to implement OPTIMAL page replacement policies.

```
#include<stdio.h>
int main()
{
   int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k, pos, max, faults = 0;
   printf("Enter number of frames: ");
   scanf("%d", &no_of_frames);

   printf("Enter number of pages: ");
   scanf("%d", &no_of_pages);

   printf("Enter page reference string: ");

   for(i = 0; i < no_of_pages; ++i){
      scanf("%d", &pages[i]);
   }

   for(i = 0; i < no_of_frames; ++i){
      frames[i] = -1;
   }

   for(i = 0; i < no_of_pages; ++i){
      flag1 = flag2 = 0;

      for(j = 0; j < no_of_frames; ++j){
         if(frames[j] == pages[i]){
               flag1 = flag2 = 1;
               break;
            }
      }

      if(flag1 == 0){
         for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == -1){
               faults++;
               frames[j] = pages[i];
               flag2 = 1;
               break;
            }
```

```
            }
        }
        if(flag2 == 0){
         flag3 =0;
            for(j = 0; j < no_of_frames; ++j){
             temp[j] = -1;

             for(k = i + 1; k < no_of_pages; ++k){
             if(frames[j] == pages[k]){
             temp[j] = k;
             break;
             }
             }
            }

            for(j = 0; j < no_of_frames; ++j){
             if(temp[j] == -1){
             pos = j;
             flag3 = 1;
             break;
             }
            }

            if(flag3 ==0){
             max = temp[0];
             pos = 0;

             for(j = 1; j < no_of_frames; ++j){
             if(temp[j] > max){
             max = temp[j];
             pos = j;
             }
             }
            }
frames[pos] = pages[i];
faults++;
        }
        printf("\n");
        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }

    printf("\n\nTotal Page Faults = %d", faults);

    return 0;
}
```

. Write a command to create a file .Find out letter ‘a’ in file

Command = grep a filename.

· Write a program to implement FCFS disk scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
 int RQ[100],i,n,TotalHeadMoment=0,initial;
 printf("Enter the number of Requests\n");
 scanf("%d",&n);

 printf("Enter the Requests sequence\n");
 for(i=0;i<n;i++)
 scanf("%d",&RQ[i]);

 printf("Enter initial head position\n");
 scanf("%d",&initial);

 // logic for FCFS disk scheduling

 for(i=0;i<n;i++)
 {
 TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
 initial=RQ[i];
 }

 printf("Total head moment is %d",TotalHeadMoment);
return 0;

 }
```