

OS PRACTICALS

2

SORT

Original contents of data.txt:

apples
oranges
pears
kiwis
bananas

*****OUTPUT*****

```
ssjcoe@LL5-comp13:~$ sort data.txt
```

apples
bananas
kiwis
oranges
pears

-----**GREP**-----

```
ssjcoe@LL5-comp13:~$ egrep -w "pears" data.txt
```

pears

AWK

Original contents of marks.txt:

- 1) Root Physics 80
- 2) Rahul Math 90
- 3) Shyam Biology 87
- 4) Kedar English 85
- 5) Hari History 89

*****OUTPUT*****

```
ssjcoe@LL5-comp13:~$ awk '{print $3 "\t" $4}' marks.txt
```

Physics 80

Math 90

Biology 87

English 85

History 89

echo "Logged in users are:-"

who -u

echo "Number of logged in users are:-"

who -u | wc -l

*****OUTPUT*****

LOGGED IN USERS ARE:-

ssjcoe :0 2019-03-16 11:22 ? 1273 (:0)

ssjcoe pts/0 2019-03-16 11:28 ? 2457 (:0)

NUMBER OF LOGGED IN USERS ARE:-

(2)

echo "Current Home Directory is:-"

whoami

echo "Current operating system type:-"

uname

echo "Current working directory is :-"

pwd

*****OUTPUT*****

Current Home Directory is:-

ssjcoe

Current operating system type:-

Linux

Current working directory is :-

/home/ssjcoe

(3)

```
echo "OS Name is:-"
```

```
uname
```

```
echo "Release number is:-"
```

```
uname -a
```

```
echo "Kernel version is:-"
```

```
uname -r
```

```
*****OUTPUT*****
```

```
OS Name is:-
```

```
Linux
```

```
Release number is:-
```

```
Linux LL5-comp13 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014
```

```
x86_64 x86_64 x86_64 GNU/Linux
```

```
Kernel version is:-
```

```
3.13.0-24-generic
```

(4)

echo "Display processes with highest memory usage:"

ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head

*****OUTPUT*****

ssjcoe@LL5-comp13:~\$ gedit highestmem.sh

ssjcoe@LL5-comp13:~\$ sh highestmem.sh

Display processes with highest memory usage

PID	PPID	CMD	%MEM	%CPU
1568	1258	compiz	7.8	1.6
958	956	nessusd -q	4.1	0.9
1783	1118	/usr/lib/evolution/evolutio	2.2	0.0
2183	1118	gedit	2.0	0.6
1659	1258	nautilus -n	1.9	0.1
1057	907	/usr/bin/X -core :0 -seat s	1.1	0.6
2100	1118	/usr/bin/unity-scope-loader	1.0	0.0
1256	1118	/usr/lib/x86_64-linux-gnu/h	1.0	0.0
2247	1118	gnome-terminal	1.0	0.2

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

int main() {
    int pid;

    pid = fork();
    if (pid < 0) {
        printf("Fork error");
    }
    else if (pid == 0) {
        printf("\nAfter fork");
        printf("\nThis is child process");
        printf("\nChild PID: %d", getpid());
        printf("\nParent PID: %d\n", getppid());
    }
    else {
        wait(NULL); // Parent waits for child to complete
        printf("\nBefore fork");
        printf("\nThis is parent process");
        printf("\nParent PID: %d", getpid());
        printf("\nChild PID: %d\n", pid);
    }
}
```

```
return 0;}
```

*****OUTPUT*****

After fork

This is child process

Child PID: 3757

Parent PID: 3756

Before fork

This is parent process

Parent PID: 3756

Child PID: 3757

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

void main() {
    int fd1, fd2;
    char buf[60];
    fd1 = open("abc.txt", O_RDWR);
    fd2 = open("pqr.txt", O_RDWR);
    read(fd1, buf, sizeof(buf));
    write(fd2, buf, sizeof(buf));
    close(fd1);
    close(fd2);
}
```

*****OUTPUT*****

Contents of abc.txt:

Hello

Contents of pqr.txt:

Hello


```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

void main() {
    int fd1, fd2;
    char buf[60];
    char *p = "abc.txt";
    fd1 = open("abc.txt", O_RDWR);
    fd2 = open("pqr.txt", O_RDWR);
    read(fd1, buf, sizeof(buf));
    write(fd2, buf, sizeof(buf));
    remove(p);
    close(fd1);
    close(fd2);
}
```

*****OUTPUT*****

Contents of pqr.txt:

Hello

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <dirent.h>

void main() {
    DIR *dp;
    struct dirent *dirp;
    dp = opendir(".");
    while ((dirp = readdir(dp)) != NULL) {
        if (dirp->d_ino == 0)
            continue;
        else
            printf("%s\n", dirp->d_name);
    }
    closedir(dp);
}

```

*****OUTPUT*****

Public

.rpm

abcd.txt
marks.txt
.Pallin.sh.swo
.rnd
fork.c~
.pallin.sh.swp
0001.pcap
at.dat
.profile
.thunderbird
Nessus-6.8.1-ubuntu110_i386.deb
cd
iptables.txt
all-2.0.tar.gz
content.c~
Desktop
cp.c
file1.txt
exp3.sh
exp7.c
examples.desktop
.config
.bashrc
.Pallin.sh.swm
highestmem.sh~
Videos
fork.c

data1.txt

.Pallin.sh.swp

5

```
#include<stdio.h>
#define MIN -9999;
struct proc
{
    int no,at,bt,ct,wt,tat,pri,status;
};
struct proc read(int i)
{
    struct proc p;
    printf("\nProcess No: %d\n",i);
    p.no=i;
    printf("Enter Arrival Time: ");
    scanf("%d",&p.at);
    printf("Enter Burst Time: ");
    scanf("%d",&p.bt);
    printf("Enter Priority: ");
    scanf("%d",&p.pri);
    p.status=0;
    return p;
}

void main()
{
    int n,l,ct=0,remaining;
    struct proc p[10],temp;
    float avgtat=0,avgwt=0;
    printf("<--Highest Priority First Scheduling Algorithm (Non-Preemptive)-->\n");
    printf("Enter Number of Processes: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
        p[i]=read(i+1);
    for(int i=0;i<n-1;i++)
        for(int j=0;j<n-i-1;j++)
            if(p[j].at>p[j+1].at)
            {
                temp=p[j];
                p[j]=p[j+1];
                p[j+1]=temp;
            }
}
```

```

    }
    p[9].pri=MIN;
    remaining=n;
    printf("\nProcessNo\tAT\tBT\tPri\tCT\tTAT\tWT\tRT\n");
    for(ct=p[0].at;remaining!=0;)
    {
        l=9;
        for(int i=0;i<n;i++)
            if(p[i].at<=ct && p[i].status!=1 && p[i].pri>p[l].pri)
                l=i;
        p[l].ct=ct+p[l].bt;
        p[l].tat=p[l].ct-p[l].at;
        avgtat+=p[l].tat;
        p[l].wt=p[l].tat-p[l].bt;
        avgwt+=p[l].wt;
        p[l].status=1;
        remaining--;

    printf("P%d\tt%d\tt%d\tt%d\tt%d\tt%d\tt%d\tt%d\n",p[l].no,p[l].at,p[l].bt,p[l].pri,p[l].ct,p[l].tat,p[l].
    wt,p[l].wt);
    }
    avgtat/=n,avgwt/=n;
    printf("\nAverage TurnAroundTime=%f\nAverage WaitingTime=%f",avgtat,avgwt);
}

```

```

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define BUFFER_SIZE 5

#define MAX_ITEMS 5

int buffer[BUFFER_SIZE];

int in = 0;

int out = 0;

int produced_count = 0;

int consumed_count = 0;

pthread_mutex_t mutex;

pthread_cond_t full;

pthread_cond_t empty;

void* producer(void* arg) {

    int item = 1;

    while (produced_count < MAX_ITEMS) {

        pthread_mutex_lock(&mutex);

        while (((in + 1) % BUFFER_SIZE) == out) {

            pthread_cond_wait(&empty, &mutex);

        }

        buffer[in] = item;

        printf("Produced: %d", item);

        item++;

        in = (in + 1) % BUFFER_SIZE;

        produced_count++;
    }
}

```

```

    pthread_cond_signal(&full);
    pthread_mutex_unlock(&mutex);
}
pthread_exit(NULL);
}

void* consumer(void* arg) {
    while (consumed_count < MAX_ITEMS) {
        pthread_mutex_lock(&mutex);
        while (in == out) {
            pthread_cond_wait(&full, &mutex);
        }
        int item = buffer[out];
        printf("Consumed: %d", item);
        out = (out + 1) % BUFFER_SIZE;
        consumed_count++;
        pthread_cond_signal(&empty);
        pthread_mutex_unlock(&mutex);
    }
    pthread_exit(NULL);
}

int main() {
    pthread_t producerThread, consumerThread;
    pthread_mutex_init(&mutex, NULL);
    pthread_cond_init(&full, NULL);
    pthread_cond_init(&empty, NULL);

    pthread_create(&producerThread, NULL, producer, NULL);

```

```
pthread_create(&consumerThread, NULL, consumer, NULL);  
pthread_join(producerThread, NULL);  
pthread_join(consumerThread, NULL);  
pthread_mutex_destroy(&mutex);  
pthread_cond_destroy(&full);  
pthread_cond_destroy(&empty);  
return 0;  
}
```

*****OUTPUT*****

```
Produced: 1  
Produced: 2  
Produced: 3  
Produced: 4  
Consumed: 1  
Consumed: 2  
Consumed: 3  
Consumed: 4  
Produced: 5  
Consumed: 5
```



```

#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<unistd.h>
sem_t chopstick[5];
void * philos(void *);
void eat(int);
int main()
{
    int i,n[5];
    pthread_t T[5];
    for(i=0;i<5;i++)
        sem_init(&chopstick[i],0,1);
    for(i=0;i<5;i++){
        n[i]=i;
        pthread_create(&T[i],NULL,philos,(void *)&n[i]);
    }
    for(i=0;i<5;i++)
        pthread_join(T[i],NULL);
}
void * philos(void * n)
{
    int ph=*(int *)n;
    printf("Philosopher %d wants to eat\n",ph);

```

```

    printf("Philosopher %d tries to pick left chopstick\n",ph);
    sem_wait(&chopstick[ph]);
    printf("Philosopher %d picks the left chopstick\n",ph);
    printf("Philosopher %d tries to pick the right chopstick\n",ph);
    sem_wait(&chopstick[(ph+1)%5]);
    printf("Philosopher %d picks the right chopstick\n",ph);
    eat(ph);
    sleep(2);
    printf("Philosopher %d has finished eating\n",ph);
    sem_post(&chopstick[(ph+1)%5]);
    printf("Philosopher %d leaves the right chopstick\n",ph);
    sem_post(&chopstick[ph]);
    printf("Philosopher %d leaves the left chopstick\n",ph);
}

void eat(int ph)
{
    printf("Philosopher %d begins to eat\n",ph);
}

```

MFT MEMORY MANAGEMENT TECHNIQUE

```
#include<stdio.h>
#include<conio.h>
main()
{
int ms, bs, nob, ef,n, mp[10],tif=0;
int i,p=0;
clrscr();
printf("Enter the total memory available (in Bytes) -- ");
scanf("%d",&ms);
printf("Enter the block size (in Bytes) -- ");
scanf("%d", &bs);
nob=ms/bs;
ef=ms - nob*bs;
printf("\nEnter the number of processes -- ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter memory required for process %d (in Bytes)-- ",i+1);
scanf("%d",&mp[i]);
}
printf("\nNo. of Blocks available in memory -- %d",nob);
printf("\n\nPROCESS\tMEMORY REQUIRED\tALLOCATED\tINTERNAL
FRAGMENTATION");
for(i=0;i<n && p<nob;i++)
{
printf("\n %d\t\t%d",i+1,mp[i]);
if(mp[i] > bs)
printf("\t\tNO\t\t---");
else
{
printf("\t\tYES\t\t%d",bs-mp[i]);
tif = tif + bs-mp[i];
p++;
}
}
if(i<n)
printf("\nMemory is Full, Remaining Processes cannot be accomodated");
printf("\n\nTotal Internal Fragmentation is %d",tif);
printf("\nTotal External Fragmentation is %d",ef);
getch();
```

}

INPUT

Enter the total memory available (in Bytes) -- 1000

Enter the block size (in Bytes)-- 300

Enter the number of processes -- 5

Enter memory required for process 1 (in Bytes) -- 275

Enter memory required for process 2 (in Bytes) -- 400

Enter memory required for process 3 (in Bytes) -- 290

Enter memory required for process 4 (in Bytes) -- 293

Enter memory required for process 5 (in Bytes) -- 100

No. of Blocks available in memory -- 3

OUTPUT

PROCESS MEMORY-REQUIRED ALLOCATED INTERNAL-FRAGMENTATION

1	275	YES	25
2	400	NO	-----
3	290	YES	10
4	293	YES	7

Memory is Full, Remaining Processes cannot be accommodated

Total Internal Fragmentation is 42

Total External Fragmentation is 100

B. MVT MEMORY MANAGEMENT TECHNIQUE

```
#include<stdio.h>
#include<conio.h>
main()
{
int ms,mp[10],i, temp,n=0;
char ch = 'y';
clrscr();
printf("\nEnter the total memory available (in Bytes)-- ");
scanf("%d",&ms);
temp=ms;
for(i=0;ch=='y';i++,n++)
{
printf("\nEnter memory required for process %d (in Bytes) -- ",i+1);
scanf("%d",&mp[i]);
if(mp[i]<=temp)
{
printf("\nMemory is allocated for Process %d ",i+1);
temp = temp - mp[i];
}
else
{
printf("\nMemory is Full");
break;
}
printf("\nDo you want to continue(y/n) -- ");
scanf(" %c", &ch);
}
printf("\n\nTotal Memory Available -- %d", ms);
printf("\n\n\tPROCESS\t\tMEMORY ALLOCATED ");
for(i=0;i<n;i++)
printf("\n \t%d\t\t\t%d",i+1,mp[i]);
printf("\n\nTotal Memory Allocated is %d",ms-temp);
printf("\nTotal External Fragmentation is %d",temp);
getch();
}
```

INPUT

```
Enter the total memory available (in Bytes) -- 1000
Enter memory required for process 1 (in Bytes) -- 400
Memory is allocated for Process 1
Do you want to continue(y/n) -- y
Enter memory required for process 2 (in Bytes) -- 275
Memory is allocated for Process 2
Do you want to continue(y/n) -- y
Enter memory required for process 3 (in Bytes) -- 550
```

OUTPUT

Memory is Full

Total Memory Available -- 1000

PROCESS MEMORY-ALLOCATED

1	400
---	-----

2	275
---	-----

Total Memory Allocated is 675

Total External Fragmentation is 325

```

#include<stdio.h>

int main()
{
    int i,j,n,a[50],frame[10],no,k,avail,count=0;
    printf("\n ENTER THE NUMBER OF PAGES:\n");
    scanf("%d",&n);
    printf("\n ENTER THE PAGE NUMBER :\n");
    for(i=1;i<=n;i++)
        scanf("%d",&a[i]);
    printf("\n ENTER THE NUMBER OF FRAMES :");
    scanf("%d",&no);
    for(i=0;i<no;i++)
        frame[i]= -1;
    j=0;
    printf("\tref string\t page frames\n");
    for(i=1;i<=n;i++)
    {
        printf("%d\t\t",a[i]);
        avail=0;
        for(k=0;k<no;k++)
            if(frame[k]==a[i])
                avail=1;
        if (avail==0)
        {
            frame[j]=a[i];
            j=(j+1)%no;
            count++;
            for(k=0;k<no;k++)
                printf("%d\t",frame[k]);
        }
        printf("\n");
    }
    printf("Page Fault Is %d",count);
    return 0;
}

```

OUTPUT:

ENTER THE NUMBER OF PAGES: 20

ENTER THE PAGE NUMBER : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

ENTER THE NUMBER OF FRAMES :3

<u>ref string</u>		<u>page frames</u>	
7	7	-1	-1
0	7	0	-1
1	7	0	1
2	2	0	1
0			
3	2	3	1
0	2	3	0
4	4	3	0
2	4	2	0
3	4	2	3
0	0	2	3
3			
2			
1	0	1	3
2	0	1	2
0			
1			
7	7	1	2
0	7	0	2
1	7	0	1

Page Fault Is 15


```

#include <stdio.h>
#include<conio.h>
void main()
{
int f[50], i, st, len, j, c, k, count = 0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
printf("Files Allocated are : \n");
x: count=0;
printf("Enter starting block and length of files: ");
scanf("%d%d", &st,&len);
for(k=st;k<(st+len);k++)
if(f[k]==0)
count++;
if(len==count)
{
for(j=st;j<(st+len);j++)
if(f[j]==0)
{
f[j]=1;
printf("%d\t%d\n",j,f[j]);
}
if(j!=(st+len-1))
printf(" The file is allocated to disk\n");
}
else
printf(" The file is not allocated \n");
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit();
getch();
}

```

Program Output:

Files Allocated are :

Enter starting block and length of files: 14 3

14 1

15 1

16 1

The file is allocated to disk

Do you want to enter more file(Yes - 1/No - 0)1

Enter starting block and length of files: 14 1

The file is not allocated

Do you want to enter more file(Yes - 1/No - 0)1

Enter starting block and length of files: 14 4

The file is not allocated

Do you want to enter more file(Yes - 1/No - 0)0