OS lab program

//systemcalls(1)

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
void main()
{
        int i,ret;
        pid_t pid;
        pid=fork();
        if(pid<0)
        {
                printf("Process Creation Error");
                exit(-1);
        }
        else if(pid>0)
        {
                wait(NULL);
                printf("\nParent Starts\nEven Numbers:");
                for(i=0;i<=10;i+=2)
                        printf("%d",i);
                printf("\nProcess id: %d",getpid());
                printf("\nParent Ends\nCalling execl()\n");
                ret=execl("/bin/ls","ls","-1",(char *)0);
        }
        else if(pid==0)
        {
                printf("\nChild Starts\nOdd Numbers:");
                for(i=1;i<=10;i+=2)
                        printf("%d",i);
                printf("\nProcess id: %d",getpid());
                printf("\nParent Process id: %d",getppid());
                printf("\nChild Ends\n");
        }
}
```

//filesystem(2)

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

```
#include<string.h>
#include<unistd.h>
#include<fcntl.h>
int main()
{
        int file;
        char rdata[50],wdata[50];
        printf("Enter a string:");
        scanf("%s",wdata);
        printf("Writing the string to file.txt");
        file=open("file.txt",O_WRONLY|O_CREAT);
        write(file,wdata,strlen(wdata));
        close(file);
        printf("\nNow, reading the current data in file.txt\n");
        file=open("file.txt",O_RDONLY);
        write(1,rdata,read(file,rdata,50));
        close(file);
        printf("\nEnter string to add to file:");
        scanf("%s",wdata);
        file=open("file.txt",O_APPEND|O_WRONLY);
         write(file,wdata,strlen(wdata));
    close(file);
    printf("Now, reading the current data in file.txt\n");
    file=open("file.txt",O_RDONLY);
    write(1,rdata,read(file,rdata,50));
    close(file);
}
```

//fcfs(3)

```
#include<stdio.h>
struct process
{
        int pid,btime,wtime,ttime;
}p[10];
void main()
{
        int i,j,k,n,tturn,twait;
        float await,aturn;
        printf("Enter the number of processes:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
```

```
{
		printf("Burst Time for Process p%d (in ms):",(i+1));
		scanf("%d",&p[i].btime);
		p[i].pid=i+1;
}
p[0].wtime=0;
for(i=0;i<n;i++)
{
		p[i+1].wtime=p[i].wtime+p[i].btime;
		p[i].ttime=p[i].wtime+p[i].btime;
}
tturn=twait=0;
for(i=0;i<n;i++)
{
		tturn=p[i].ttime;
		twait+=p[i].wtime;
}
await=(float)twait/n;
aturn=(float)tturn/n;
printf("\n FCFS scheduling\n-");
for(i=0;i<(p[n-1].ttime+2*n);i++)
		printf("-");
printf("\n Gantt Chart\n");
printf("-");
for(i=0;i<(p[n-1].ttime+2*n);i++)
 printf("-");
printf("\n");
printf("|");
for(i=0;i<n;i++)
{
		k=p[i].btime/2;
		for(j=0;j<k;j++)
			printf(" ");
		printf("P%i",p[i].pid);
		for(j=+1;j<p[i].btime;j++)
			printf(" ");
		printf("|");
}
printf("\n");
printf("-");
for(i=0;i<(p[n-1].ttime+2*n);i++)
```

```
                printf("-");
        printf("\n");
        printf("0");
        for(i=0;i<n;i++)
        {
                for(j=0;j<p[i].btime;j++)
                        printf(" ");
                printf("%2d",p[i].ttime);
        }
        printf("\n\nProcess ID\tBurst Time\tTurnAroundTime\tWait Time\n");
        for(i=0;i<n;i++)
                printf("\np%d\t\t%d\t\t%d\t\t%d",p[i].pid,p[i].btime,p[i].ttime,p[i].wtime);
        printf("\nAverage Waiting Time:%5.2fms",await);
        printf("\nAverage Turn Around Time:%5.2fms",aturn);
}
```

//sjf(4)

```
#include<stdio.h>
struct process
{
int pid;
int btime;
int wtime;
int ttime;
}p[10],temp;
void main()
{
int i,j,k,n,tturn,twait;
float await,aturn;
printf("\nenter the number of process");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nBurst time for theprocess p%d(in ms)",(i+1));
scanf("%d",&p[i].btime);
p[i].pid=i+1;
}
p[0].wtime=0;
for(i=0;i<(n-1);i++)
{
        for(j=i+1;j<n;j++)
```

```
		{
		if((p[i].btime>p[j].btime)||((p[i].btime)==p[j].btime&&p[i].pid>p[j].pid))
		{
			temp=p[i];
			p[i]=p[j];
			p[j]=temp;
		}
		}
}
for(i=0;i<n;i++)
{
p[i+1].wtime=p[i].wtime+p[i].btime;
p[i].ttime=p[i].wtime+p[i].btime;
}
tturn=twait=0;
for(i=0;i<n;i++)
{
tturn+=p[i].ttime;
twait+=p[i].wtime;
}
await=(float)twait/n;
aturn=(float)tturn/n;
printf("\nSJF scheduling\n \n");
for(i=0;i<(p[n-1].ttime+2*n);i++)
printf("-");
printf("\n\n GANTT CHART \n");
printf("-");
for(i=0;i<(p[n-1].ttime+2*n);i++)
printf("-");
printf("\n|")
for(i=0;i<n;i++)
{
		k=p[i].btime/2;
		for(j=0;j<k;j++)
		printf(" ");
		printf("p %d",p[i].pid);
		for(j=+1;j<p[i].btime;j++)
		printf(" ");
		printf("|");
}
printf("\n");
```

```
printf("-");
for(i=0;i<(p[n-1].ttime+2*n);i++)
printf("-");
printf("\n0");
for(i=0;i<n;i++)
{
		for(j=0;j<p[i].btime;j++)
		printf(" ");
		printf("%2d",p[i].ttime);
}

printf("\nprocess\t \t b-time\t\t t-time \t\t w-time\n");
for(i=0;i<n;i++)
printf("\np%d\t%d\t%d\t%d",p[i].pid,p[i].btime,p[i].ttime,p[i].wtime);
printf("\n \n Average waiting:%5.2fms",await);
printf("\n \n Average turn around time:%5.2fms",aturn);
}
```

//priority(5)

```
#include<stdio.h>
struct process
{
		int pid,btime,wtime,ttime,priority;
}p[10],temp;
void main()
{
		int i,j,k,n,tturn,twait;
		float await,aturn;
		printf("Enter the number of processes:");
		scanf("%d",&n);
		for(i=0;i<n;i++)
		{
				printf("Burst Time for Process p%d (in ms):",(i+1));
				scanf("%d",&p[i].btime);
				printf("Priority for process p%d:",i+1);
				scanf("%d",&p[i].priority);
				p[i].pid=i+1;
		}
		p[0].wtime=0;
		for(i=0;i<n-1;i++)
	{
```

```
for(j=i+1;j<n;j++)
{
      if(p[i].priority>p[j].priority)
      {
            temp=p[i];
            p[i]=p[j];
            p[j]=temp;
      }
}


      }
      for(i=0;i<n;i++)
      {
                  p[i+1].wtime=p[i].wtime+p[i].btime;
                  p[i].ttime=p[i].wtime+p[i].btime;
      }
      tturn=twait=0;
      for(i=0;i<n;i++)
      {
                  tturn+=p[i].ttime;
                  twait+=p[i].wtime;
      }
      await=(float)twait/n;
      aturn=(float)tturn/n;
      printf("\n Priority Scheduling\n-");
      for(i=0;i<(p[n-1].ttime+2*n);i++)
                  printf("-");
      printf("\n Gantt Chart\n");
      printf("-");
      for(i=0;i<(p[n-1].ttime+2*n);i++)
printf("-");
      printf("\n");
      printf("|");
      for(i=0;i<n;i++)
      {
                  k=p[i].btime/2;
                  for(j=0;j<k;j++)
                              printf(" ");
                  printf("P%i",p[i].pid);
                  for(j=+1;j<p[i].btime;j++)
                              printf(" ");
```

```
                printf("|");
        }
        printf("\n");
        printf("-");
        for(i=0;i<(p[n-1].ttime+2*n);i++)
                printf("-");
        printf("\n");
        printf("0");
        for(i=0;i<n;i++)
        {
                for(j=0;j<p[i].btime;j++)
                        printf(" ");
                printf("%2d",p[i].ttime);
        }
        printf("\n\nProcess ID\tPriority\tBurst Time\tTurnAroundTime\tWait Time\n");
        for(i=0;i<n;i++)
                printf("\np%d\t\t%d\t\t%d\t\t%d\t\t%d",p[i].pid,p[i].priority,p[i].btime,p[i].ttime,p[i].wtime);
        printf("\nAverage Waiting Time:%5.2fms",await);
        printf("\nAverage Turn Around Time:%5.2fms",aturn);
}
```

//fib(6)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
void main()
{
   pid_t pid;//Stores the process ID returned by fork()
   int pfd[2];//Array to hold file descriptors for the pipe.
   int i,j,flag,f1,f2,f3,n;
   static unsigned int ar[50],br[50];
   printf("Enter the value of n:  ");
   scanf("%d", &n);
   if(pipe(pfd)==-1)//Creates a pipe
   {
      printf("Error in pipe");
      exit(-1);
   }
   pid = fork();//create a new process
```

```
//CHILD PROCESS
if (pid == 0)
{
   printf("\nChild process generates fib series:\n");
   f1=-1;
   f2=+1;
   for(i=0;i<n;i++)
   {
      f3=f1+f2;
      printf("%d\t",f3);
      ar[i]=f3;
      f1=f2;
      f2=f3;
   }
   write(pfd[1],ar,n*sizeof(int));
}
//PARENT PROCESS
else if (pid > 0)
{
   wait(NULL);
   read(pfd[0],br,n*sizeof(int));
   printf("\nParent print fib number that are also prime \n");
   for(i=0;i<n;i++)
   {
      flag=0;
      if(br[i]<=1)
      flag=1;
      for(j=2;j<=br[i]/2;j++)
      {
         if(br[i]%j==0)
         {
            flag=1;
            break;
         }
      }
      if(flag==0)
      printf("%d\t",br[i]);
   }printf("\n");
}
else
{
```

```
        printf("Process creation failed");
        exit(-1);
    }
}
```

//ipc

Server.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/un.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#define shmsize 27
void main()
{
        char c;
        int shmid;
        key_t key=2013;
        char *shm,*s;
        if((shmid=shmget(key,shmsize,IPC_CREAT|0666))<0)
        {
                perror("shmget");
                exit(1);
        }
        printf("Shared Memory ID:%d\n",shmid);
        if((shm=shmat(shmid,NULL,0))==(char*)-1)
        {
                perror("shmat");
                exit(1);
        }
        memset(shm,0,shmsize);
        s=shm;
        printf("Writing (a-z) onto shared memory\n");
        for(c='a';c<='z';c++)
                *s++=c;
        *s='\0';
        while(*shm!='*');
        printf("Client Finished Reading\n");
        if(shmdt(shm)!=0)
                fprintf(stderr,"could not close memory segment.\n");
```

```
	shmctl(shmid,IPC_RMID,0);
}
```

Client.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#define shmsize 27
void main()
{
	int shmid;
	key_t key=2013;
	char *shm,*s;
	if((shmid=shmget(key,shmsize,0666))<0)
	{
		printf("Server not started\n");
		exit(1);
	}
	else
		printf("Accessing Shared Memory ID:%d\n",shmid);
	if((shm=shmat(shmid,NULL,0))==(char*)-1)
	{
		perror("shmat");
		exit(1);
	}
	printf("shared memory contents\n");
	for(s=shm;*s!='\0';s++)
		putchar(*s);
	putchar('\n');
	*shm='*';
}
```

//best(8)

```
#include<stdio.h>
struct process
{
	int size;
	int flag;
	int holeid;
}p[10];
```

```
struct hole
{
        int size,hid;
        int actual;
}h[10];
void main()
{
        int i,np,nh,j;
        void bsort(struct hole[],int);
        printf("\nEnter the number of holes:");
        scanf("%d",&nh);
        for(i=0;i<nh;i++)
        {
                printf("\nEnter size for hole H%d:",i);
                scanf("%d",&h[i].size);
                h[i].actual=h[i].size;
                h[i].hid=i;
        }
        printf("\nEnter number of process:");
        scanf("%d",&np);
        for(i=0;i<np;i++)
        {
                printf("\nEnter size of process p%d:",i);
                scanf("%d",&p[i].size);
                p[i].flag=0;
        }
        for(i=0;i<np;i++)
        {
                bsort(h,nh);
                for(j=0;j<nh;j++)
                {
                        if(p[i].flag!=1)
                        {
                                if(p[i].size<=h[j].size)
                                {
                                        p[i].flag=1;
                                        p[i].holeid=h[j].hid;
                                        h[j].size-=p[i].size;
                                }
                        }
                }
```

```
		}
		printf("\n\t Best Fit\n");
		printf("\nprocess\tpsize\thole");
		for(i=0;i<=np;i++)
		{
			if(p[i].flag!=1)
				printf("\np%d\t%d\tNot Allocate ",i,p[i].size);
			else
				printf("\np%d\t%d\tH%d",i,p[i].size,p[i].holeid);

		}
		printf("\n\nHole\tactual\tAvailable");
		for(i=0;i<nh;i++)
			printf("\nH%d\t%d\t%d",h[i].hid,h[i].actual,h[i].size);
		printf("\n");
}
void  bsort(struct hole bh[],int n)
{
		struct hole temp;
		int i,j;
		for(i=0;i<n-1;i++)
		{
			for(j=i+1;j<n;j++)
			{
				if(bh[i].size>bh[j].size)
				{
					temp=bh[i];
					bh[i]=bh[j];
					bh[j]=temp;
				}
			}
		}
}
```

//first(9)

```
#include<stdio.h>
struct process
{
		int size;
		int flag;
		int holeid;
```

```
}p[10];
struct hole
{
        int size;
        int actual;
}h[10];
void main()
{
        int i,np,nh,j;
        printf("\nEnter the number of holes:");
        scanf("%d",&nh);
        for(i=0;i<nh;i++)
        {
                printf("\nEnter size for hole H%d:",i);
                scanf("%d",&h[i].size);
                h[i].actual=h[i].size;
        }
        printf("\nEnter number of process:");
        scanf("%d",&np);
        for(i=0;i<np;i++)
        {
                printf("\nEnter size of process p%d:",i);
                scanf("%d",&p[i].size);
                p[i].flag=0;
        }
        for(i=0;i<np;i++)
        {
                for(j=0;j<nh;j++)
                {
                        if(p[i].flag!=1)
                        {
                                p[i].flag=1;
                                p[i].holeid=i;
                                h[j].size-=p[i].size;
                        }
                }
        }
        printf("\n\t First Fit\n");
        printf("\nprocess\tpsize\thole");
        for(i=0;i<np;i++)
        {
```

```
                if(p[i].flag!=1)
                        printf("\np%d\t%d\tNot Allocate ",i,p[i].size);
                else
                        printf("\np%d\t%d\tH%d",i,p[i].size,p[i].holeid);

        }
        printf("\n\nHole\tactual\tAvailable");
        for(i=0;i<nh;i++)
                printf("\nH%d\t%d\t%d",i,h[i].actual,h[i].size);
        printf("\n");
}
```

//contiguous allocation

```
#include<stdio.h>
#include<string.h>
int num=0,length[10],start[10];
char fid[20][4],a[20][4];
void directory()
{
        int i;
        printf("\nFile Start Length\n");
        for(i=0;i<num;i++)
                printf("%-4s %3d %6d \n",fid[i],start[i],length[i]);
}
void display()
{
        int i;
        for(i=0;i<20;i++)
                printf("%4d",i);
        printf("\n");
        for(i=0;i<20;i++)
                printf("%4s",a[i]);
}
void main()
{
        int i,n,k,temp,st,nb,ch,flag;
        char id[4];
        for(i=0;i<20;i++)
                strcpy(a[i]," ");
        printf("Disk space before allocation:\n");
        display();
```

```
	do
	{
		printf("\nEnter file name(max 3 char):\n");
		scanf("%s",id);
		printf("Enter start block:");
		scanf("%d",&st);
		printf("\nEnter number of blocks:");
		scanf("%d",&nb);
		strcpy(fid[num],id);
		length[num]=nb;
		flag=0;
		if((st+nb)>20)
		{
			printf("\nRequirement exceeds range\n");
			continue;
		}
		for(i=st;i<(st+nb);i++)
			if(strcmp(a[i]," ")!=0)
				flag=1;
		if(flag==1)
		{
			printf("\nContiguous allocation not possible\n");
			continue;
		}
		start[num]=st;
		for(i=st;i<(st+nb);i++)
			strcpy(a[i],id);
		printf("\nAllocation Done");
		num++;
		printf("\nAny more allocation?(y/n):");
		scanf("%d",&ch);
	}while(ch==1);
	printf("\n\t\tContiguous Allocation\n");
	printf("\nDirectory:");
	directory();
	printf("\nDisk space after allocation:\n");
	display();
	printf("\n");
}
```

//procon(11)

```
#include<stdio.h>
#include<stdlib.h>
int mutex=1,full=0,empty,x=0;
int main()
{
	int n;
	void producer();
	void consumer();
	int wait(int);
	int signal(int);
	printf("Enter buffer size:");
	scanf("%d",&empty);
	printf("\n1.Producer\n2.Consumer\n3.Exit");
	while(1)
	{
		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;
		}
	}
	return 0;
}
int wait(int s)
{
	return (--s);
}
int signal(int s)
```

```
{
        return(++s);
}
void producer()
{
        mutex=wait(mutex);
        full=signal(full);
        empty=wait(empty);
        x++;
        printf("\nProducer produces the item %d",x);
        mutex=signal(mutex);
}
void consumer()
{
        mutex=wait(mutex);
        full=wait(full);
        empty=signal(empty);
        printf("\nConsumer consumes item %d",x);
        x--;
        mutex=signal(mutex);
}
```