**PART-A**

**(OPERATING SYSTEM LAB)**

**EXPERIMENT NO: 1**

**NAME OF THE EXPERIMENT:** Write a programs using the following system calls of UNIX operating system: a) fork, b) exec, c) getpid, d) exit, e) wait, f)close, g)stat, h)opendir & readdir.

**Program:**

**a) fork() system call**

#include<stdio.h>

main()

{

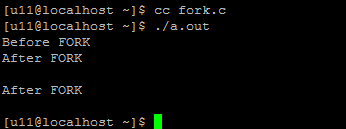
printf("Before FORK \n");

fork();

printf("After FORK \n\n");

}

**Output:**

****

**b) exec() system call: transforms executable binary files in to a process.**

#include<sys/types.h>

#include<stdio.h>

include<unistd.h>

main()

{

exec("/bin/ls","/bin/ls");

}

**c) getpid() system call.**

#include<stdio.h>

int main()

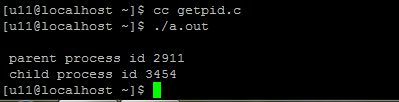
{

printf("\n parent process id %d",getppid());

printf("\n child process id %d\n",getpid());

}

**Output:**

****

**d) exit(): system call: this system call ends a process and returns a value to its parent.**

#include<sys/types.h>

#include<stdio.h>

include<unistd.h>

main()

{

int p=fork();

if(p==0)

{

printf("/n child created");

exit(0);

printf("process ended");

}

if(p<0)

{

printf("can't create child");

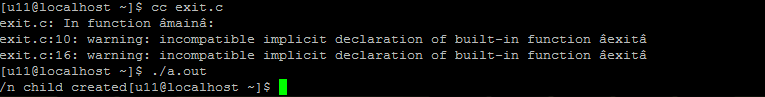
exit(-1);

printf("process ended");

}

}

**Output:**



**e) wait(): we can control the execution of a child process to by calling wait() in the parent process.**

wait() forces the parent process to suspend execution until the child completes. It returns the process id of a child that have finished.

#include<sys/types.h>

#include<stdio.h>

include<unistd.h>

main()

{

int pid,i=0;

printf("ready to fork");

pid=fork();

if(pid==0)

{

printf("child starts");

for(i=0;i<10;i++)

{

printf("child ends");

}//for

}//if

else

{

wait(0);

for(i=0;i<10;i++)

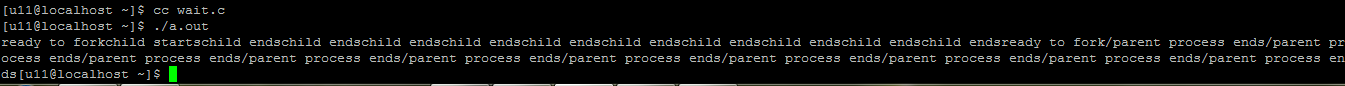
{

printf("/parent process ends");

}

}//else }

**Output:**



**f) open(),write() and close()**

#include<fcntl.h>

#include<unistd.h>

main()

{

int fd,i;

fd=open("test",O\_CREAT/O\_RDWR/O-APPEND);

for(i=0;i<10;i++)

write(fd,"CVSR",1)

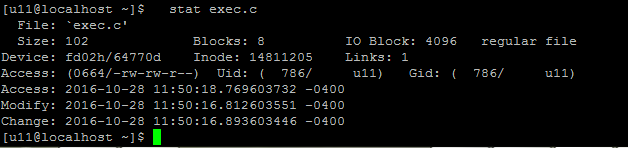
close(fd);

}

**g) stat(): this a command line system call returns the inode information of a file.**

ex: $stat filename

**Output:**



**h) opendir() and readdir()**

#include<stdio.h>

#include<sys/types.h>

#include<dirent.h>

int main(int argc,char \*argv[])

{

DIR \*dp;

struct dirent \*dirp;

if(argc!=2)

{

printf("a single argument (the dir name) is required");

exit(1);

}

if((dp=opendir(argv[1]))==NULL)

{

printf("cant open%s".argv[1]);

exit(1);

}

while((dirp=readdir(dp))!=NULL)

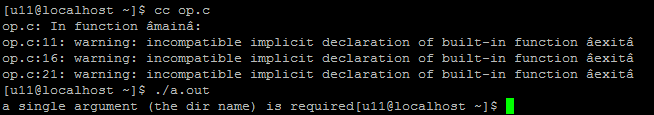
printf("%s %d\n",dirp->d\_name,dirp->d\_ino);

closedir(dp);

exit(0);

}

**Output:**



**EXPERIMENT NO: 2**

**NAME OF THE EXPERIMENT:** Write a program to implement multithreading?

**Program:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include<pthread.h>

void \*thread\_fun1(void \*arg);

void \*thread\_fun2(void \*arg);

void \*thread\_fun3(void \*arg);s

char message[]="welcome to linux";

int main()

{

pthread\_t a\_thread1;

pthread\_t a\_thread2;

pthread\_t a\_thread3;

void \*thread\_result1;

void \*thread\_result2;

void \*thread\_result3;

pthread\_create(&a\_thread1,NULL,thread\_fun1,(void \*)message);

pthread\_create(&a\_thread2,NULL,thread\_fun2,(void \*)message);

pthread\_create(&a\_thread3,NULL,thread\_fun3,(void \*)message);

printf("waitin for thread to finish \n");

pthread\_join(a\_thread1,&thread\_result1);

pthread\_join(a\_thread2,&thread\_result2);

pthread\_join(a\_thread3,&thread\_result3);

printf("thread joined, it returned %s\n",(char\*)thread\_result1);

printf("thread joined, it returned %s\n",(char\*)thread\_result2);

printf("thread joined, it returned %s\n",(char\*)thread\_result3);

printf("message is now %s\n",message);

printf("message is now %s\n",message);

printf("message is now %s\n",message);

exit(0);

}

void \*thread\_fun1(void \*arg)

{

printf("thread\_function is running .Argument was %s\n",(char\*)arg);

sleep(6);

strcpy(message,"bye1!");

pthread\_exit("thank you thread 1");

}

void \*thread\_fun2(void \*arg)

{

sleep(7);

printf("thread\_function is running .Argument was %s\n",(char\*)arg);

sleep(7);

strcpy(message,"bye2!");

pthread\_exit("thank you thread 2");

}

void \*thread\_fun3(void \*arg)

{

printf("thread\_function is running .Argument was %s\n",(char\*)arg);

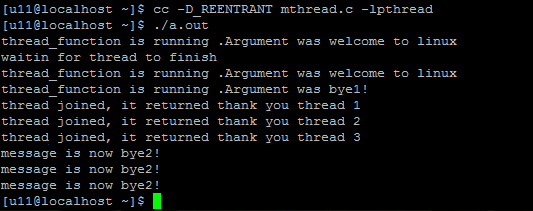
sleep(10);

strcpy(message,"bye3!");

pthread\_exit("thank you thread 3");

}

**Output:**



**EXPERIMENT NO: 3**

**NAME OF THE EXPERIMENT:** Give the list of processes, their CPU burst times and arrival times, display or print the Gantt chart for FCFS and SJF. For each of the scheduling policy compute and print the average waiting time and average turnaround time.

**Program: a) FCFS:**

#include<stdio.h>

main()

{

int bt[20], wt[20], tat[20], i, n;

float wtavg, tatavg;

printf("\nEnter the number of processes -- ");

scanf("%d", &n);

for(i=0;i<n;i++)

{

printf("\nEnter Burst Time for Process %d -- ", i);

scanf("%d", &bt[i]);

}

wt[0] = wtavg = 0;

tat[0] = tatavg = bt[0];

for(i=1;i<n;i++)

{

wt[i] = wt[i-1] +bt[i-1];

tat[i] = tat[i-1] +bt[i];

wtavg = wtavg + wt[i];

tatavg = tatavg + tat[i];

}

printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");

for(i=0;i<n;i++)

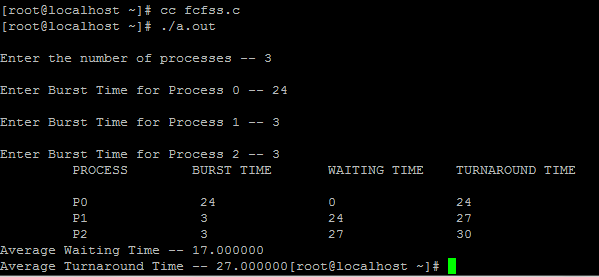
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);

printf("\nAverage Waiting Time -- %f", wtavg/n);

printf("\nAverage Turnaround Time -- %f", tatavg/n);

}

**Output:**

****

**b) SJF:**

#include<stdio.h>

main()

{

int p[20], bt[20], wt[20], tat[20], i, k, n, temp;

float wtavg, tatavg;

printf("\nEnter the number of processes -- ");

scanf("%d", &n);

for(i=0;i<n;i++)

{

p[i]=i;

printf("Enter Burst Time for Process %d -- ", i);

scanf("%d", &bt[i]);

}

for(i=0;i<n;i++)

for(k=i+1;k<n;k++)

if(bt[i]>bt[k])

{

temp=bt[i];

bt[i]=bt[k];

bt[k]=temp;

temp=p[i];

p[i]=p[k];

p[k]=temp;

}

wt[0] = wtavg = 0;

tat[0] = tatavg = bt[0];

for(i=1;i<n;i++)

{

wt[i] = wt[i-1] +bt[i-1];

tat[i] = tat[i-1] +bt[i];

wtavg = wtavg + wt[i];

tatavg = tatavg + tat[i];

}

printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");

for(i=0;i<n;i++)

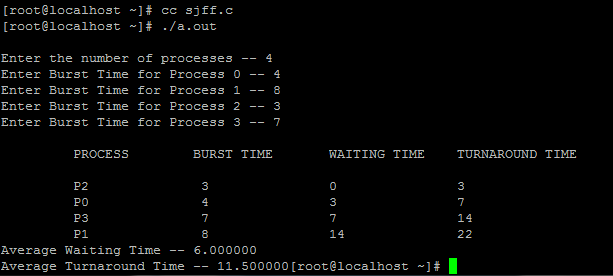
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);

printf("\nAverage Waiting Time -- %f", wtavg/n);

printf("\nAverage Turnaround Time -- %f", tatavg/n);

}

**Output:**

****

**EXPERIMENT NO: 4**

**NAME OF THE EXPERIMENT:** Give the list of processes, their CPU burst times and arrival times, display or print the Gantt chart for Priority and Round Rabin. For each of the scheduling policy compute and print the average waiting time and average turnaround time.

**Program:**

1. **Priority Scheduling:**

#include<stdio.h>

main()

{

int p[20],bt[20],pri[20], wt[20],tat[20],i, k, n, temp;

float wtavg, tatavg;

printf("Enter the number of processes --- ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

p[i] = i;

printf("Enter the Burst Time & Priority of Process %d --- ",i);

scanf("%d %d",&bt[i], &pri[i]);

}

for(i=0;i<n;i++)

for(k=i+1;k<n;k++)

if(pri[i] > pri[k])

{

temp=p[i];

p[i]=p[k];

p[k]=temp;

temp=bt[i];

bt[i]=bt[k];

bt[k]=temp;

temp=pri[i];

pri[i]=pri[k];

pri[k]=temp;

}

wtavg = wt[0] = 0;

tatavg = tat[0] = bt[0];

for(i=1;i<n;i++)

{

wt[i] = wt[i-1] + bt[i-1];

tat[i] = tat[i-1] + bt[i];

wtavg = wtavg + wt[i];

tatavg = tatavg + tat[i];

}

printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND TIME");

for(i=0;i<n;i++)

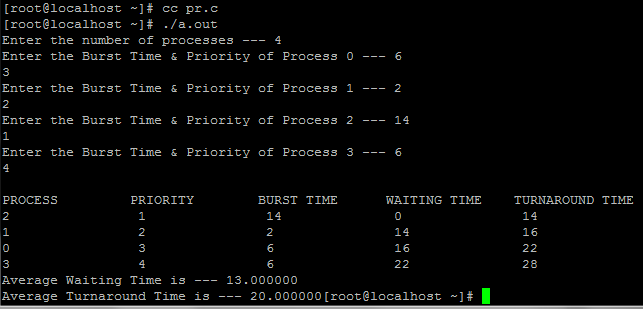
printf("\n%d \t\t %d \t\t %d \t\t %d \t\t %d ",p[i],pri[i],bt[i],wt[i],tat[i]);

printf("\nAverage Waiting Time is --- %f",wtavg/n);

printf("\nAverage Turnaround Time is --- %f",tatavg/n);

}

**Output:**

****

1. **Round Robin Scheduling:**

#include<stdio.h>

main()

{

int n,bt[10],tq,i,et=0,tbt=0,c[10],b[10],twt=0,w[10],count=0;

float avgwt,avgtt;

printf("Enetr the number of processes:");

scanf("%d",&n);

printf("Enter %d burst times:",n);

for(i=0;i<n;i++)

scanf("%d",&bt[i]);

printf("Enter time quantum:");

scanf("%d",&tq);

for(i=0;i<n;i++)

{

b[i]=0;

w[i]=0;

c[i]=0;

tbt=tbt+bt[i];

}

for(i=0;count!=n;)

{

if(bt[i]<0)

{

printf("error");

}

else if(bt[i]>0)

{

if(bt[i]>tq)

{

bt[i]=bt[i]-tq;

et=et+tq;

b[i]=b[i]+tq;

}

else

{

et=et+bt[i];

b[i]=b[i]+bt[i];

bt[i]=0;

c[i]=et;

count++;

}

}

i++;

if(i>=n)

i=0;

}

for(i=0;i<n;i++)

{

w[i]=c[i]-b[i];

twt=twt+w[i];

}

avgtt=(float)(tbt+twt)/n;

avgwt=(float)twt/n;

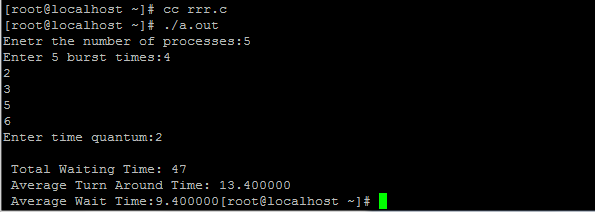
printf("\n Total Waiting Time: %d",twt);

printf("\n Average Turn Around Time: %f",avgtt);

printf("\n Average Wait Time:%f",avgwt);

}

**Output:**

****

**EXPERIMENT NO: 5**

**NAME OF THE EXPERIMENT:** Implement producer consumer problem using semaphore?

**Program:**

#include<stdio.h>

#define BUFFERSIZE 10

int mutex,n,empty,full=0,item,item1;

int buffer[20];

int in=0,out=0,mutex=1;

void wait(int s)

{

while(s<0)

{

printf(“\nCannot add an item\n”);

exit(0);

}

s--;

}

void signal(int s)

{

s++;

}

void producer()

{

do

{

wait (empty);

wait(mutex);

printf(“\nEnter an item:”); scanf(“%d”,&item);

buffer[in]=item;

in=in+1;

signal(mutex);

signal(full);

}

while(in<n);

}

void consumer()

{

do

{

wait(full);

wait(mutex);

item1=buffer[out];

printf(“\nConsumed item =%d”,item1);

out=out+1;

signal(mutex);

signal(empty);

}

while(out<n);

}

void main()

{

printf(“Enter the value of n:”);

scanf(“%d “,&n);

empty=n;

while(in<n)

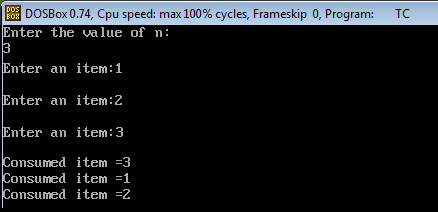
producer();

while(in!=out)

consumer();

}

**Output:**

****

**EXPERIMENT NO: 6**

**NAME OF THE EXPERIMENT:** Write a program to implement Banker’s algorithm for deadlock avoidance?

**Program:**

#include<stdio.h>

void main()

{

int k=0,output[10],d=0,t=0,ins[5],i,avail[5],allocated[10] [5],need[10][5],MAX[10][5],pno,P[10],j,rz, count=0;

printf("\n Enter the number of resources : ");

scanf("%d", &rz);

printf("\n enter the max instances of each resources\n");

for (i=0;i<rz;i++)

{

avail[i]=0;

printf("%c= ",(i+97));

scanf("%d",&ins[i]);

}

printf("\n Enter the number of processes : ");

scanf("%d", &pno);

printf("\n Enter the allocation matrix \n ");

for (i=0;i<rz;i++)

printf(" %c",(i+97));

printf("\n");

for (i=0;i <pno;i++)

{

P[i]=i;

printf("P[%d] ",P[i]);

for (j=0;j<rz;j++)

{

scanf("%d",&allocated[i][j]);

avail[j]+=allocated[i][j];

}

}

printf("\nEnter the MAX matrix \n ");

for (i=0;i<rz;i++)

{

printf(" %c",(i+97));

avail[i]=ins[i]-avail[i];

}

printf("\n");

for (i=0;i <pno;i++)

{

printf("P[%d] ",i);

for (j=0;j<rz;j++)

scanf("%d", &MAX[i][j]);

}

printf("\n");

A: d=-1;

for (i=0;i <pno;i++)

{

count=0;

t=P[i];

for (j=0;j<rz;j++)

{

need[t][j] = MAX[t][j]-allocated[t][j];

if(need[t][j]<=avail[j])

count++;

}

if(count==rz)

{

output[k++]=P[i];

for (j=0;j<rz;j++)

avail[j]+=allocated[t][j];

}

else

P[++d]=P[i];

}

if(d!=-1)

{

pno=d+1;

goto A;

}

printf("\t <");

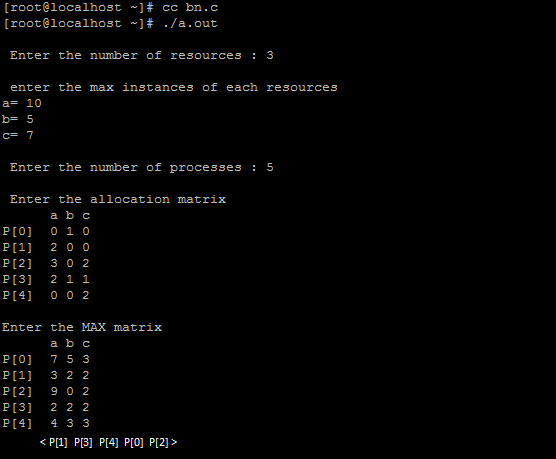
for (i=0;i<k;i++)

printf(" P[%d] ",output[i]);

printf(">");

}

**Output:**



**EXPERIMENT NO: 7**

**NAME OF THE EXPERIMENT:** Write a program to implement page replacement algorithms (FCFS, Optimal, LRU)

**a) FIFO:**

**AIM:** A program to simulate FIFO Page Replacement Algorithm

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{

int a[5],b[20],n,p=0,q=0,m=0,h,k,i,q1=1;

char f='F';

clrscr();

printf("Enter the Number of Pages:");

scanf("%d",&n);

printf("Enter %d Page Numbers:",n);

for(i=0;i<n;i++)

scanf("%d",&b[i]);

for(i=0;i<n;i++)

{

if(p==0)

{

if(q>=3)

q=0;

a[q]=b[i];

q++;

if(q1<3)

{

q1=q;

}

}

printf("\n%d",b[i]);

printf("\t");

for(h=0;h<q1;h++)

printf("%d",a[h]);

if((p==0)&&(q<=3))

{

printf("-->%c",f);

m++;

}

p=0;

for(k=0;k<q1;k++)

{

if(b[i+1]==a[k])

p=1;

}

}

printf("\nNo of faults:%d",m);

}

**Output:**

**Input:**

Enter the Number of Pages: 12

Enter 12 Page Numbers:

2 3 2 1 5 2 4 5 3 2 5 2

**Output:**

2 2-> F

3 23-> F

2 23

1 231-> F

5 531-> F

2 521-> F

4 524-> F

5 524

3 324-> F

2 324

5 354-> F

2 352-> F

No of faults: 9

**b) Optimal :**

**AIM: A program to implement optimal Page Replacement Algorithm**

**Program:**

#include<stdio.h>

main()

{

int a[50],b[3],n,i,j,count,rep,k;

printf("enter the total no of pages:\n");

scanf("%d",&n);

printf("eneter the page references \n");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

for(i=0;i<3;i++)

{

b[i]=a[i];

printf("%d ",b[i]);

printf("\n");

}

for(i=0;i<3;i++)

printf("%d ",b[i]);

count=3;

for(i=3;i<n;i++)

{

k=0;

while(k<3)

{

if(b[k]==a[i])

break;

else

k++;

}

if(k==3)

{

rep=find(a,b,i);

b[rep]=a[i];

count++;

}

for(j=0;j<3;j++)

printf("%d ",b[j]);

printf("\n");

}

printf("page faults %d \n",count);

}

int find(int m[],int n[],int pos)

{

int p1,p2,p3,loc,var;

p1=p2=p3=var=0;

loc=pos;

while(n[0]!=m[pos])

pos++;

p1=pos;

pos=loc;

while(n[1]!=m[pos])

pos++;

p2=pos;

pos=loc;

while(n[2]!=m[pos])

pos++;

p3=pos;

pos=loc;

if(p1>p2&&p1>p3)

var=0;

else if(p2>p1&&p2>p3)

var=1;

else if(p3>p1&&p3>p2)

var=2;

return(var);

}

**Output:**

enter the total no of pages:

10

enter the page references

2

3

4

1

2

3

1

2

3

0

2

3

4

2 3 4 2 3 1

2 3 1

2 3 1

2 3 1

2 3 1

2 3 1

0 3 1

page faults 5

1. **LRU:**

**AIM: A program to implement LRU Page Replacement Algorithm**

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{

int g=0,a[5],b[20],p=0,q=0,m=0,h,k,i,q1=1,j,u,n;

char f='F';

clrscr();

printf("Enter the number of pages:");

scanf("%d",&n);

printf("Enter %d Page Numbers:",n);

for(i=0;i<n;i++)

scanf("%d",&b[i]);

for(i=0;i<n;i++)

{

if(p==0)

{

if(q>=3) // frame size

q=0;

a[q]=b[i];

q++;

if(q1<3)

{

q1=q;

//g=1;

}

}

printf("\n%d",b[i]);

printf("\t");

for(h=0;h<q1;h++)

printf("%d",a[h]);

if((p==0)&&(q<=3))

{

printf("-->%c",f);

m++;

}

p=0;

g=0;

if(q1==3)

{

for(k=0;k<q1;k++)

{

if(b[i+1]==a[k])

p=1;

}

for(j=0;j<q1;j++)

{

u=0;

k=i;

while(k>=(i-1)&&(k>=0))

{

if(b[k]==a[j])

u++;

k--;

}

if(u==0)

q=j;

}

}

else

{

for(k=0;k<q;k++)

{

if(b[i+1]==a[k])

p=1;

}

}

}

printf("\nNo of faults:%d",m);

getch();

}

**Output:**

**Input:**

Enter the Number of Pages: 12

Enter 12 Page Numbers:

2 3 2 1 5 2 4 5 3 2 5 2

**Output:**

2 2-> F

3 23-> F

2 23

1 231-> F

5 251-> F

2 251

4 254-> F

5 254

3 354-> F

2 352-> F

5 352

2 352

No of faults: 7

**PART-B**

**(COMPUTER NETWORKS LAB)**

**EXPERIMENT NO: 1**

**NAME OF THE EXPERIMENT:** Implement the data link layer framing methods Bit stuffing, Character Stuffing.

a)**BIT STUFFING**

**Aim:** Implement the data link layer framing methods such as Bit Stuffing.

**Theory:** Security and Error detection are the most prominent features that are to be provided by any application which transfers data from one end to the other end. One of such a mechanism in tracking errors which may add up to the original data during transfer is known as Stuffing. It is of two types namely Bit Stuffing and the other Character Stuffing. Coming to the Bit Stuffing, 01111110 is appended within the original data while transfer of it. The following program describes how it is stuffed at the sender end and de-stuffed at the receiver end.

**Program:**  
#include<stdio.h>

main()

{

int a[15];

int i,j,k,n,c=0,pos=0;

clrscr();

printf("\n Enter the frame length:");

scanf("%d",&n);

printf("\n Enter input frame (0’s & 1’s only):");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

for(i=0;i<n;i++)

{

if(a[i]==1)

{

c++;

if(c==5)

{

pos=i+1;

c=0;

for(j=n;j>=pos;j--)

{

k=j+1;

a[k]=a[j];

}

a[pos]=0;

n=n+1;

}}

else

c=0;

}

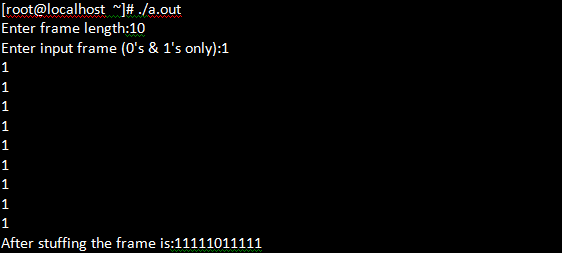
printf("\n DATA AFTER STUFFING \n");

for(i=0;i<n;i++)

printf("%d",a[i]);

getch();

}

**Output:**

**b) CHARACTER STUFFING:**

**Aim**: Implement the data link layer framing methods such as Character Stuffing and also De-stuff it  
**Theory:** Coming to the Character Stuffing, DLESTX and DLEETX are used to denote start and end of character data with some constraints imposed on repetition of characters as shown in the program below clearly.

**Program:**  
  
#include<stdio.h>

#include<conio.h>

#include<string.h>

void charc(void);

void main()

{

int choice;

while(1)

{

printf("\n\n\n1.character stuffing");

printf("\n\n2.exit");

printf("\n\n\n enter choice");

scanf("%d", &choice);

printf("%d", choice);

if(choice>2)

printf("\n\n invalid option....please renter");

switch(choice)

{

case 1: charc();

break;

case 2: exit(0);

}}}

void charc(void)

{

char c[50],d[50],t[50];

int i,m,j;

clrscr();

printf("enter the number of characters\n");

scanf("%d",&m);

printf("\n enter the characters\n");

for(i=0;i<m;i++)

scanf("%c",&c[i]);

printf("\n original data\n");

for(i=0;i<m;i++)

printf("%c",c[i]);

d[0]='d';

d[1]='l';

d[2]='e';

d[3]='s';

d[4]='t';

d[5]='x';

for(i=0,j=6;i<m;i++)

{

if((c[i]=='d'&&c[i+1]=='l'&& c[i+2]=='e'))

{

d[j]='d';

j++;

d[j]='l';

j++;

d[j]='e';

j++;

m=m+3;

}

d[j]=c[i];

}

m=m+6;

m++;

d[m]='d';

m++;

d[m]='l';

m++;

d[m]='e';

m++;

d[m]='e';

m++;

d[m]='t';

m++;

d[m]='x';

m++;

printf("\n\n transmitted data: \n");

for(i=0;i {

printf("%c",d[i]);

}

for(i=6,j=0;i <m;i++)

{

if(d[i]=='d'&&d[i+1]=='l'&&d[i+2]=='e'&&d[i+3]=='d'&&d[i+4]=='l'&&d[i+5]=='e')

i=i+3;

t[j]=d[i];

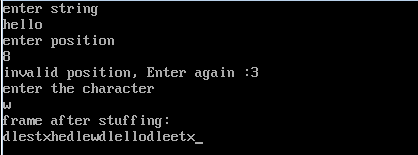
}

printf("\n\nreceived data:");

for(i=0;i {printf("%c",t[i]);

}

}  
**Output:**

****

**EXPERIMENT NO: 2**

**NAME OF THE EXPERIMENT:**  Implement CRC 16 error control mechanism in data link layer

**AIM:** Implement CRC 16 error control mechanism in data link layer

**THEORY:** CRC means Cyclic Redundancy Check. It is the most famous and traditionally successful mechanism used in error detection through the parity bits installed within the data and obtaining checksum which acts as the verifier to check whether the data retreived at the reciever end is genuine or not. Various operations are involved in implementing CRC on a data set through CRC generating polynomials. In the program, I have also provided the user to opt for Error detection whereby he can proceed for it. Understand the program below as it is much simpler than pretented to be so.

**SOURCE CODE:**

#include <stdio.h>

#include <string.h>

void main()

{

int i,j,keylen,msglen;

char input[100], key[30],temp[30],quot[100],rem[30],key1[30];

printf("Enter Data: ");

gets(input);

printf("Enter Key: ");

gets(key);

keylen=strlen(key);

msglen=strlen(input);

strcpy(key1,key);

for(i=0;i<keylen-1;i++)

{

input[msglen+i]='0';

}

for(i=0;i<keylen;i++)

temp[i]=input[i];

for(i=0;i<msglen;i++)

{

quot[i]=temp[0];

if(quot[i]=='0')

for(j=0;j<keylen;j++)

key[j]='0';

else

for(j=0;j<keylen;j++)

key[j]=key1[j];

for(j=keylen-1;j>0;j--)

{

if(temp[j]==key[j])

rem[j-1]='0';

else

rem[j-1]='1';

}

rem[keylen-1]=input[i+keylen];

strcpy(temp,rem);

}

strcpy(rem,temp);

printf("\nQuotient is ");

for(i=0;i<msglen;i++)

printf("%c",quot[i]);

printf("\nRemainder is ");

for(i=0;i<keylen-1;i++)

printf("%c",rem[i]);

printf("\nFinal data is: ");

for(i=0;i<msglen;i++)

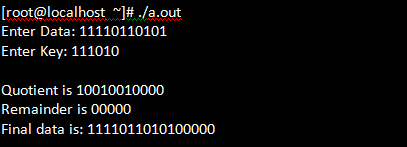
printf("%c",input[i]);

for(i=0;i<keylen-1;i++)

printf("%c",rem[i]);

}

**OUTPUT:**



**EXPERIMENT NO: 3**

**NAME OF THE EXPERIMENT:** Implement minimum hamming Distance

**AIM:** To implement minimum hamming Distance.

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

char data[5];

int encoded[8],edata[7],syndrome[3];

int hmatrix[3][7] = {

1,0,0,0,1,1,1,

0,1,0,1,0,1,1,

0,0,1,1,1,0,1

};

char gmatrix[4][8]={"0111000","1010100","1100010","1110001"};

int main(){

int i,j;

system("clear");

printf("\nHamming code----- Encoding\n");

printf("Enter 4 bit data : ");

scanf("%s",data);

printf("\nGenerator matrix\n");

for(i=0;i<4;i++)

printf("%s\n",gmatrix[i]);

printf("\nEncoded data ");

for(i=0;i<7;i++)

{

for(j=0;j<4;j++)

encoded[i]+=((data[j]-'0')\*(gmatrix[j][i]-'0'));

encoded[i]=encoded[i]%2;

printf("%d ",encoded[i]);

}

printf("\nHamming code----- Decoding\n");

printf("Enter encoded bits as received : ");

for(i=0;i<7;i++)

scanf("%d",&edata[i]);

for(i=0;i<3;i++)

{

for(j=0;j<7;j++)

syndrome[i]+=(edata[j]\*hmatrix[i][j]);

syndrome[i]=syndrome[i]%2;

}

for(j=0;j<7;j++)

if((syndrome[0]==hmatrix[0][j]) && (syndrome[1]==hmatrix[1][j])&& (syndrome[2]==hmatrix[2][j]))

break;

if(j==7)

printf("\nError free\n");

else

{

printf("\nError received at bit number %d of data\n",j+1);

edata[j]=!edata[j];

printf("\nCorrect data should be : ");

for(i=0;i<7;i++)

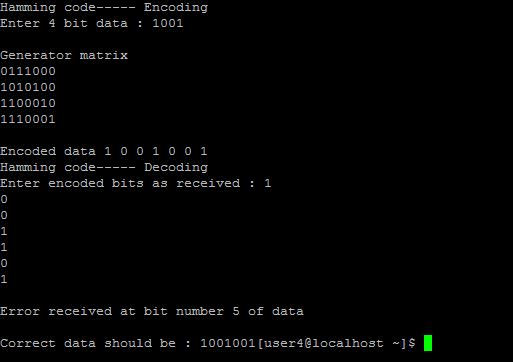
printf("%d",edata[i]);

}

return 0;

}

**OUTPUT:**



**EXPERIMENT NO: 4**

**NAME OF THE EXPERIMENT:** Implement Stop and Wait protocol.

**AIM:** Implement Stop and Wait protocol.

**SOURCE CODE:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include<pthread.h>

#include<semaphore.h>

void \*thread\_fun(void \*arg);

sem\_t bin\_sem;

char work\_area[30];

int main()

{

int res;

pthread\_t a\_thread;

void \*thread\_result;

res=sem\_init(&bin\_sem,0,0);

res=pthread\_create(&a\_thread,NULL,thread\_fun,NULL);

printf("\n Sender please send data:\n Enter 'END' for Terminate Connection\n");

while(strncmp("end",work\_area,3)!=0)

{

fgets(work\_area,30,stdin);

sem\_post(&bin\_sem);

}

printf("\n Waiting for Receiver to Finish\n");

res=pthread\_join(a\_thread,&thread\_result);

printf("\n Receiver Completed\n");

sem\_destroy(&bin\_sem);

//return 0;

exit(EXIT\_SUCCESS);

}

void \*thread\_fun(void \*arg)

{

sem\_wait(&bin\_sem);

while(strncmp("end",work\_area,3)!=0)

{

printf("receiver received data is %s %d characters\n",&work\_area,strlen(work\_area)-1);

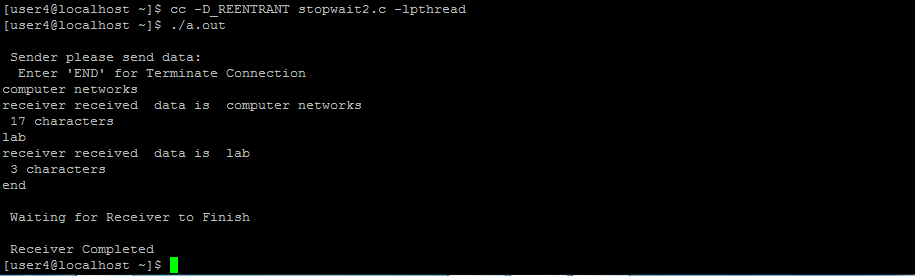
sem\_wait(&bin\_sem);

}

pthread\_exit(NULL);

}

**OUTPUT:**



**EXPERIMENT NO: 5**

**NAME OF THE EXPERIMENT:** Implement Go-Back-N and Selective Repeat Request protocols.

**AIM:** Implement Go-Back-N and Selective Repeat Request protocols.

**SOURCE CODE:**

**//Implement Go-Back-N.**

#include<stdio.h>

#include<string.h>

void main()

{

int i=1;

int j=0;

int l,k=1;

int static s;

char x,t=1;

char str1[10];

char str2[20];

char str3[10];

printf("\n NOTE NOTE Data representatin is a=1,b=2......if u are sending characters\n");

printf("\n Sender Please Enter data:");

scanf("%s",str1);

x=str1[0];

printf("\n first character in sender data is:%c\n",x);

l=strlen(str1);

for(j=0;j<20;j++)

str2[j]='0';

str2[j]='\0';

while(k)

{

for(s=0,i=0,j=0;str1[i];i++,j++,x++)

{

if(str1[i]==x)

{

str2[j]=str1[i];

s=j;

}

else

continue;

}

for(i=0;str2[i];i++)

{

if(str2[i]=='0')

{

printf("\n sender please send the data from %d",++i);

break;

}

else

k=0;

}

}

printf("\n Enter missing data:");

scanf("%s",str3);

for(s=s+1,k=0;str3[k];k++,s++)

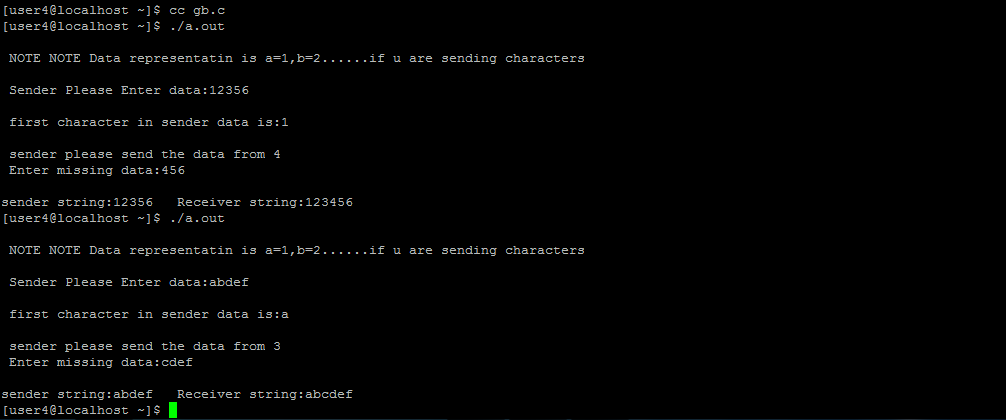
str2[s]=str3[k];

str2[s]='\0';

printf("\nsender string:%s Receiver string:%s\n",str1,str2);

}

**Output:**

****

**SOURCE CODE:**

**//Implement Selective Repeat Request protocols.**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char str1[20];

char str2[20];

int i=0;

int j=0;

char x;

//int t;

char c[2];

char ch;

clrscr();

printf("\n NOTE NOTE THE PROGRAM IS SELECTIVE REPEAT \n");

printf("\n MEANS U CAN MISS ONE BYTE \n");

printf("\n SENDER RAGHAVA ENTER THE STRING:");

scanf("%s",str1);

for(i=0,x=str1[0];str1[i];i++,x++)

{

if(str1[i]==x)

continue;

else

{

printf("\n RAGHAVA Please Enter the Missing Character:%c",x);

break;

}

}

x=str1[0];

for(i=0,j=0;str1[i];i++,j++,x++)

{

if(str1[i]==x)

str2[j]=str1[i];

else

{

str2[j]='0';

i--;

}

}

str2[j]='\0';

for(i=0;str2[i];i++)

{

if(str2[i]!='0')

continue;

else

{

printf("\n Enter the missing character:");

scanf("%s",c);

c[1]='\0';

str2[i]=c[0];

}

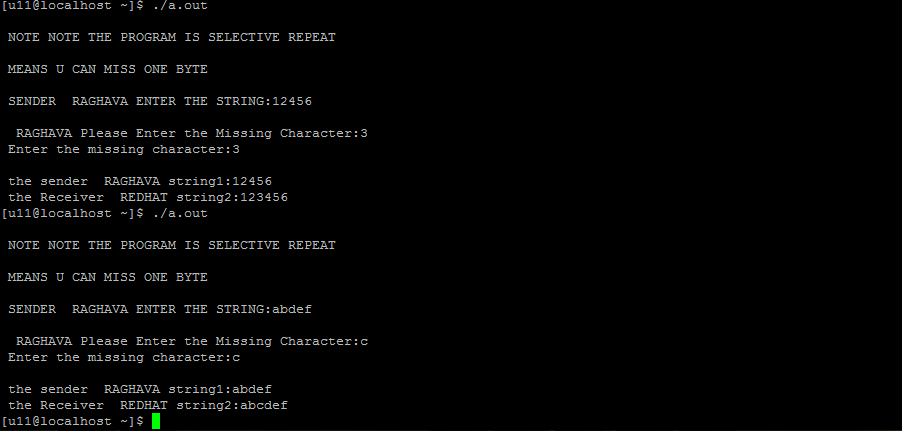
}

printf("\n the sender RAGHAVA string1:%s \n the Receiver REDHAT string2:%s\n",str1,str2);

getch();

}

**Output:**

****

**EXPERIMENT NO: 6**

**NAME OF THE EXPERIMENT:** Implement CSMA/CD Using C program.

**AIM:** To implement CSMA/CD Using C program.

**SOURCE CODE:**

// IMPLEMENTATION OF CSMA/CD

#include<sys/time.h>

int x=0;

void capture()

{

exit(0);

}

int get()

{

return x;

}

void put()

{

x++;

}

void node(char \*p)

{

int name;

int seq1,seq2,i=0;

long int t;

struct timeval tv;

struct timezone tz;

name=atoi(p);

while(1)

{

seq1=get();

seq2=get();

if(seq1==seq2)

{

put();

seq1=get();

printf("station %d transmitting frame %d\n",name,++i);

sleep(3);

seq2=get();

if(seq1!=seq2)

{

printf("station %d collision occured %d \n",name,i--);

sleep(3);

}

else

{

printf("station %d complete\n",name,i);

}

}

sleep(3);

}

}

main()

{

pthread\_t t1,t2,t3;

pthread\_create(&t1,0,(void \*)node,"1");

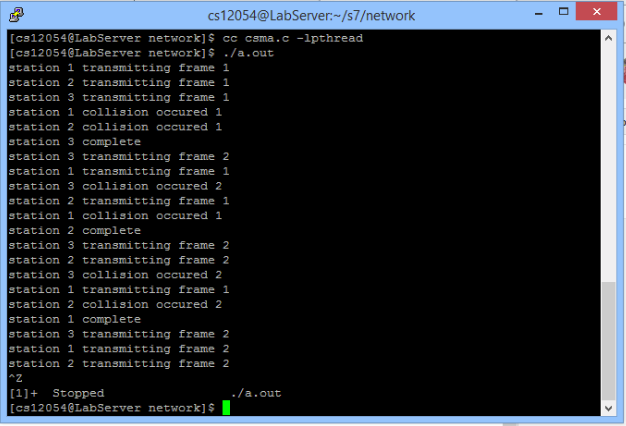
pthread\_create(&t2,0,(void \*)node,"2");

pthread\_create(&t3,0,(void \*)node,"3");

while(1);

}

**OUTPUT:**

[](http://3.bp.blogspot.com/-GelJXzcGvP8/VZpKlBabY_I/AAAAAAAAANE/GzUq5RfJEr4/s1600/Capture.PNG)

**EXPERIMENT NO: 7**

**NAME OF THE EXPERIMENT:** Write a C program for CSMA/CA.

**AIM:** Write a C program for CSMA/CA.

**ALGORITHM/FLOWCHART:**

**SOURCE CODE:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include<pthread.h>

#define threads 3

void \*thread\_fun(void \*arg);

int main()

{

int res;

pthread\_t a\_thread[threads];

void \*thread\_res;

int lt;

int tt;

//printf("\n Enter the number of stations u want:");

//scanf("%d",tt);;

for(lt=0;lt<threads;lt++)

{

res = pthread\_create(&(a\_thread[lt]),NULL,thread\_fun,(void \*)lt);

if(res!=0)

{

perror("thread creation failed");

exit(EXIT\_FAILURE);

}

}

printf("Waiting for stations to finish..........\n");

for(lt=threads-1;lt>=0;lt--)

{

res = pthread\_join(a\_thread[lt],&thread\_res);

if(res==0)

printf("\n station got the channel");

else

perror("\nstations failed DUE TO COLLISSION to get channel");

}

printf("\n .....all stations done\n");

exit(EXIT\_SUCCESS);

}

void \*thread\_fun(void \*arg)

{

int my\_number=(int)arg; //convert stations channel

int r\_num;

printf("\n Station is aaccessing the channel statins is %d\n",my\_number);

r\_num = 1 + (int)(9.0\*rand()/(RAND\_MAX+1.0));

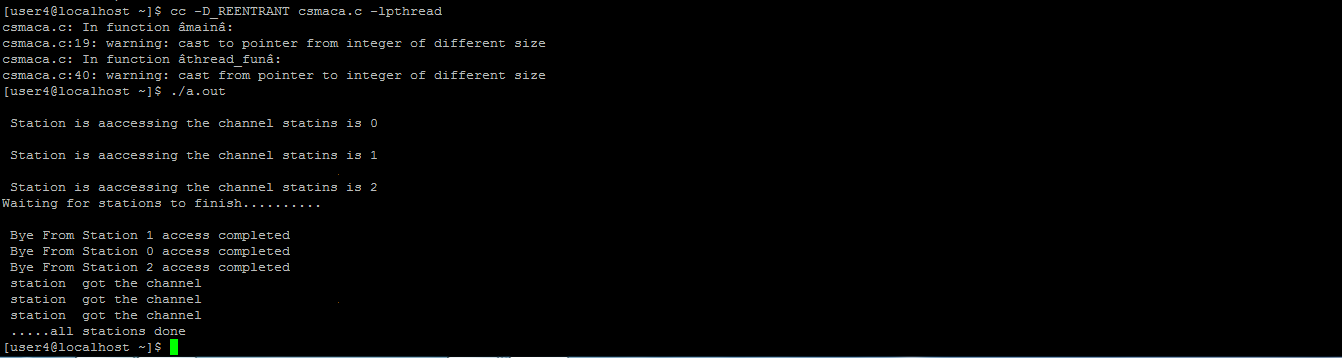
sleep(r\_num);

printf("\n Bye From Station %d access completed",my\_number);

pthread\_exit(NULL);

}

**OUTPUT:**

****