**Shell Script to convert number into word**

echo " Enter a number"

read digit

echo " Your number $digit in words"

case $digit in

0)echo "zero";;

1)echo "one";;

2)echo "two";;

3)echo "three";;

4)echo "four";;

5)echo "five";;

6)echo "six";;

7)echo "seven";;

8)echo "eight";;

9)echo "nine";;

esac

**Output:**

**Enter a number**

**1**

**Your number 1 in words**

**One**

**Shell Script to execute Arithmetic Operations:**

clear

ch='Y'

while [ $ch='y' -o $ch='Y' ]

do

echo "enter the values of a:\c"

read a

echo "enter the value of b:\c"

read b;

echo "\n 1.addition"

echo "\n 2.subraction"

echo "\n 3.multiplication"

echo "\n 4.division"

echo "\n 5.exit"

echo "enter chioce(1 2 3 4 5):\c"

read ch;

case $ch in

1) c=`expr $a + $b`

echo " the sum is="$c;;

2) c=`expr $a - $b`

echo "the sub is ="$c;;

3) c=`expr $a \\* $b`

echo "the product is="$c;;

4) c=`expr $a / $b`

echo "the div is="$c;;

5)exit;;

esac

echo "Do you want to take another choice(Y/N)"

read ch

done

echo "end of the program"

**Output:**

**enter the values of a:\c**

**3**

**enter the value of b:\c**

**3**

**\n 1.addition**

**\n 2.subraction**

**\n 3.multiplication**

**\n 4.division**

**\n 5.exit**

**enter chioce(1 2 3 4 5):\c**

**3**

**the product is=9**

**Do you want to take another choice(Y/N)**

**Shell Script to Identify given Character is in Uppercase or Lowercase**

clear

echo "enter a character:"

read ch

case $ch in

[[:upper:]])echo "enter character is in the upper case";;

[[:lower:]])echo "enter character is in the lower case";;

esac

**Output:**

**enter a character:\c**

**A**

**enter character is in the upper case**

**Shell Script to search the given Pattern in a file**

clear

while [ $choice='y' -o $choice='Y' ]

do

echo "1.basic search"

echo "2.ignore upper search result"

echo "3.enter word to search with line no"

echo "4.dispaly how many lines contain the search pattern"

echo "5.exit"

echo "enter your choice"

read choice

case $choice in

1)echo "enter word"

read word

echo "enter filename"

read filename

grep $word $filename;;

2)echo "enter word to serach lowercase or uppercase"

read word

echo "enter filename"

read filename

grep -i $word $filename;;

3)echo "enter word to search with line no"

read word

echo "enter file"

read filename

grep -n $word $filename;;

4)echo "enter word to search with how many lines thhat word is in"

read word

echo "enter file name"

read filename

grep -c $word $filename;;

5)exit;;

esac

echo "Do you want to take another choice(Y/N)"

read choice

done

echo "end of the program"

**Output:**

**1. basic search**

**2 .ignore upper search result**

**3. enter word to search with line no**

**4. Display how many lines contain the search pattern**

**5. exit**

**Enter your choice**

**2**

**Enter word to search lowercase or uppercase**

**unix**

**enter filename**

**sample.ext**

**this is line 1 UNIX UNIX**

**this is line 2 unix**

**this is line 3 Unix Unix**

**this is line 5 unix**

**Do you want to take another choice(Y/N)**

**y**

**1.basic search**

**2.ignore upper serach result**

**3.enter word to serach with line no**

**4.dispaly how many lines contain the serach pattern**

**5.exit**

**enter your choice**

**4**

**enter word to search with how many lines that word is in**

**UNIX**

**enter file name**

**sample.ext**

**1**

**Do you want to take another choice(Y/N)**

**Shell Script to find largest of three numbers**

clear

echo "enter any three values"

read a b c

if [ $a -gt $b -a $a -gt $c ]

then

echo "biggest number is a.its value is $a"

elif [ $b -gt $a -a $b -gt $c ]

then

echo "biggest number is b.its value is $b"

else

echo "\n biggest is c. its value is $c:"

fi

**Output:**

**enter any three values**

**6 7 8**

**\n biggest is c. its value is 8:**

**Shell Script to generate Fibonacci series**

clear

st=1

ft=1

ct=2

echo $ft

echo $st

while [ $ct -le 10 ]

do

ctr=`expr $ft + $st`

echo "$ctr"

ft=$st

st=$ctr

ct=`expr $ct + 1`

done

**Output:**

**1**

**1**

**2**

**3**

**5**

**8**

**13**

**21**

**34**

**55**

**89**

**Shell Script to implement Unix commands (ls, grep,wc)**

clear

ch='Y'

while [ $ch='y' -o $ch='Y' ]

do

echo "\n1.ls"

echo "\n2.grep"

echo "\n3.wc"

echo "\n4.exit"

echo "enter choice(1 2 3 4):\c"

read ch

case $ch in

1)echo "programs in your directory"

ls;;

2)echo "enter the word to search"

read word

echo "enter filename"

read filename

grep $word $filename;;

3)echo "enter file name"

read filename

wc $filename;;

4)exit;;

esac

echo"Do you want to take another choice(Y/N)"

read ch

done

echo "end of the program"

**Output:**

**\n1.ls**

**\n2.grep**

**\n3.wc**

**\n4.exit**

**enter choice(1 2 3 4):\c**

**2**

**enter the word to search**

**unix**

**enter filename**

**sample.txt**

**this is line 2 unix**

**this is line 5 unix**

**Program using process related system calls**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

main()

{

int pid,pid1,pid2;

pid=fork();

if(pid==-1)

{

printf("ERROR IN PROCESS CREATION \n");

exit(1);

}

if(pid!=0)

{

pid1=getpid();

printf("\n The parent process ID is %d\n", pid1);

}

else

{

pid2=getpid();

printf("\n The child process ID is %d\n", pid2);

}

}

**Output:**

**The child process ID is 7320**

**The parent process ID is 7319**

**Program to implement FCFS scheduling algorithm**

#include <stdio.h>

int main()

{

int n,exeTime[100],i,wTime=0,tat=0;

float awt=0,atat=0;

printf("Enter number of processes");

scanf("%d",&n);

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

{

printf("Enter exe time for process %d:",i+1);

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

}

printf("\n\n pid \t\t BT \t\t wt \t\t TAT");

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

{

tat=exeTime[i] + wTime;

printf("\n %d\t\t %d\t\t %d\t\t %d",i+1,exeTime[i],wTime,tat);

awt=awt+wTime;

atat=atat+tat;

wTime=wTime+exeTime[i];

}

awt=awt/n;

atat=atat/n;

printf("\n Average waiting time: %f",awt);

printf("\n Average turnaround time : %f",atat);

}

**Output:**

**Enter number of processes 3**

**Enter exe time for process 1:5**

**Enter exe time for process 2:9**

**Enter exe time for process 3:2**

**pid BT wt TAT**

**1 5 0 5**

**2 9 5 14**

**3 2 14 16**

**Average waiting time: 6.333333**

**Average turnaround time : 11.666667**

**Program to implement SJF scheduling algorithm**

#include<stdio.h>

int main()

{

int n,exeTime[100],wTime=0,tat=0;

int i,j,temp;

float awt=0,atat=0;

printf("enter total No.of processes");

scanf("%d",&n);

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

{

printf("enter burst time of process %d:",i+1);

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

}

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

{

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

{

if(exeTime[i]<exeTime[j])

{

temp=exeTime[i];

exeTime[i]=exeTime[j];

exeTime[j]=temp;

}

}

}

printf("\npid \t\t Bt \t\t wt \t\t tat");

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

{

tat=exeTime[i] + wTime;

printf("\n %d \t\t %d \t\t %d\t\t %d",i,exeTime[i],wTime,tat);

awt=awt+wTime;

atat=atat+tat;

wTime=wTime+exeTime[i];

}

awt=awt/n;

atat=atat/n;

printf("\n average waiting time : %f",awt);

printf("\average turnaround time:%f",atat);

}

**Output:**

**enter total No.of processes3**

**enter burst time of process 1:25**

**enter burst time of process 2:4**

**enter burst time of process 3:2**

**pid Bt wt tat**

**0 2 0 2**

**1 4 2 6**

**2 25 6 31**

**average waiting time : 2.666667verage turnaround time:13.000000**

**Program to implement producer and Consumer problem using Semaphores**

#include<stdio.h>

#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;

int main()

{

int n;

void producer();

void consumer();

int wait(int);

int signal(int);

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("\n consunmer consumes item %d",x);

x--;

mutex=signal(mutex);

}

**Output:**

**1.producer**

**2.consumer**

**3.exit**

**Enter your choice: 1**

**producer produces the item 1**

**Enter your choice:1**

**producer produces the item 2**

**Enter your choice:1**

**producer produces the item 3**

**Enter your choice:1**

**Buffer is full**

**Enter your choice:2**

**Consumer consumes item 3**

**Enter your choice:2**

**Consumer consumes item 2**

**Enter your choice:2**

**Consumer consumes item 1**

**Enter your choice:2**

**Buffer is empty**

**Program to implement Inter process communication using Pipes**

**Writer Process**

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

int main()

{

int fd;

char \* myfifo = "/tmp/myfifo";

mkfifo(myfifo, 0666);

fd = open(myfifo, O\_WRONLY);

write(fd, "Hello",sizeof("Hello"));

close(fd);

unlink(myfifo);

return 0;

}

**Reader Process**

#include <fcntl.h>

#include <stdio.h>

#include <sys/stat.h>

#include <unistd.h>

#define MAX\_BUF 1024

int main()

{

int fd;

char \* myfifo = "/tmp/myfifo";

char buf[MAX\_BUF];

/\* open, read, and display the message from the FIFO \*/

fd = open(myfifo, O\_RDONLY);

read(fd, buf, MAX\_BUF);

printf("Received: %s\n", buf);

close(fd);

return 0;

}

**Program to implement FIFO Page replacement algorithm.**

#include<stdio.h>

int main()

{

int i,j,n,a[50],frame[10],no,k,avail,count=0;

printf("\nEnter 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**

**ref string       page frames**

**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**