FCFS

import java.util.\*;

import java.lang.\*;

class Fcfs

{

public static void main (String args[])

{

Scanner s=new Scanner(System.in);

int p;

float t1=0,t2=0;

System.out.println("Enter number of processor");

p=s.nextInt();

int bt[]=new int[p];

System.out.println("Enter burst time");

for(int i=0;i<p;i++)

{

System.out.println("Enter burst time for p"+(i+1)+"=");

bt[i]=s.nextInt();

}

int wt[]=new int[p];

wt[0]=0;

for(int i=1;i<p;i++)

{

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

t1+=wt[i];

}

int tat[]=new int[p];

for(int i=0;i<p;i++)

{

tat[i]=bt[i]+wt[i];

t2+=tat[i];

}

System.out.println("process\t burst time\t twaiting time\t turnaround time");

for(int i=0;i<p;i++)

{

System.out.println("p"+(i+1)+"\t\t"+bt[i]+"\t\t"+wt[i]+"\t\t"+tat[i]);

float awt,att;

System.out.println("average waiting time="+t1/p+"ms");

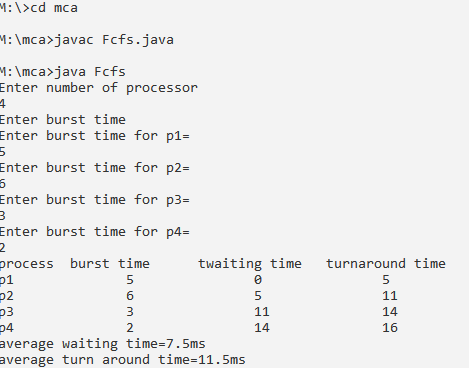
System.out.println("average turn around time="+t2/p+"ms");

}

}

}

**OUTPUT:**



SFJS

import java.util.\*;

import java.util.\*;

import java.lang.\*;

class Sjfs

{

public static void main(String args [])

{

Scanner s=new Scanner(System.in);

int p,temp,i,j,ord[];

float t1=0,t2=0;

System.out.println("enter number of processes");

p=s.nextInt();

int bt[]=new int[p];

ord=new int[p];

System.out.println("Enter burst time");

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

{

ord[i]=i+1;

bt[i]=s.nextInt();

}

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

{

for(j=i+1;j<p;j++)

{

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

{

temp=bt[i];

bt[i]=bt[j];

bt[j]=temp;

temp=ord[i];

ord[i] = ord[j];

ord[j]=temp;

}

}

}

int wt[]=new int[p];

wt[0]=0;

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

{

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

t1+=wt[i];

}

int tat[]=new int[p];

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

{

tat[i]=bt[i]+wt[i];

t2+=tat[i];

}

System.out.println("process \t burst time \t twaiting time \t turnaround time");

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

{

System.out.println("p"+ord[i]+"\t\t" +bt[i]+"\t\t"+wt[i]+"\t\t"+tat[i]);

}

float awt,att;

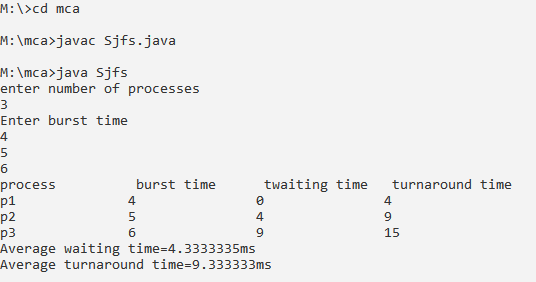
System.out.println("Average waiting time="+t1/p+"ms");

System.out.println("Average turnaround time="+t2/p+"ms");

}

}

OUTPUT**:**



PRIORITY

import java.util.\*;

import java.lang.\*;

class Priority

{

public static void main(String args[])

{

Scanner s=new Scanner(System.in);

int p,temp,i,j,order[];

float t1=0,t2=0;

System.out.println("enter no of processes");

p=s.nextInt();

int bt[]=new int[p];

order=new int[p];

int pr[]=new int[p];

System.out.println("enter burst time");

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

{

order[i]=i+1;

bt[i]=i+1;

}

System.out.println("enter priority");

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

{

order[i]=i+1;

pr[i]=s.nextInt();

}

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

{

for(j=i+1;j<p;j++)

{

if(pr[i]>pr[j])

{

temp=pr[i];

pr[j]=pr[j];

temp=bt[i];

bt[i]=bt[j];

bt[j]=temp;

temp=order[i];

order[i]=order[j];

order[j]=temp;

}

}

}

int wt[]=new int[p];

wt[0]=0;

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

{

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

t1+=wt[i];

}

int tat[]=new int[p];

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

{

tat[i]=bt[i]+wt[i];

t2+=tat[i];

}

System.out.println("process\t burst time\t priority\t waiting time \t turnaround time");

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

{

System.out.println("p"+order[i]+"\t\t"+bt[i]+"\t\t"+pr[i]+"\t\t"+wt[i]+"\t\t"+tat[i]);

}

float awt,att;

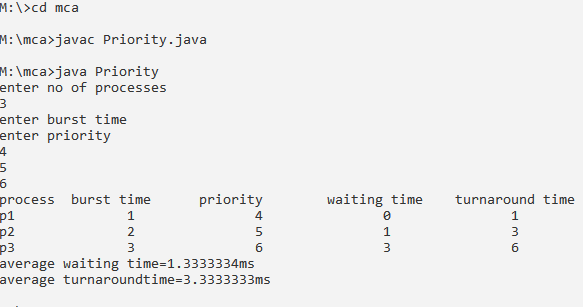
System.out.println("average waiting time="+t1/p+"ms");

System.out.println("average turnaroundtime="+t2/p+"ms");

}

}

OUTPUT:



ROUND ROBIN

import java.util.\*;

import java.lang.\*;

class RoundRobin

{

Scanner sc=new Scanner(System.in);

int[] bur,rem,wai,ta;

int size,q,b=0,t=0,flag=0;

RoundRobin(int size)

{

this.size=size;

bur=new int[size];

wai=new int[size];

ta=new int[size];

rem=new int[size];

}

void get()

{

for(int i=0;i<size;i++)

{

System.out.println("enter burst time of p"+(i+1)+":");

bur[i]=rem[i]=sc.nextInt();

}

System.out.println("enter quantum time:");

q=sc.nextInt();

}

void round()

{

do

{

flag=0;

for(int i=0;i<size;i++)

{

if(rem[i]>=q)

{

System.out.println("p"+(i+1)+"->");

for(int j=0;j<size;j++)

{

if(j==i)

rem[i]=rem[i]-q;

else if(rem[j]>0)

wai[j]+=q;

}

}

else if(rem[i]>0)

{

System.out.println("p"+(i+1)+"\t");

for(int j=0;j<size;j++)

if(j==i)

rem[i]=0;

else if(rem[j]>0)

wai[j]+=rem[i];

}

}

for(int i=0;i<size;i++)

if(rem[i]>0)

flag=1;

}

while(flag==1);

for(int i=0;i<size;i++)

ta[i]=wai[i]+bur[i];

}

void display()

{

System.out.println("\n process \tburst\twaiting\tturnaround");

for(int i=0;i<size;i++)

{

System.out.println("\t p"+(i+1)+"\t"+bur[i]+"\t"+wai[i]+"\t"+ta[i]);

b=wai[i];

t+=ta[i];

}

System.out.println("Average waiting time:"+(b/size));

System.out.println("Average turnaround time:"+(t/size));

}

}

class KRoundRobin

{

public static void main(String args[])

{

Scanner s=new Scanner(System.in);

System.out.println("Enter no of process:");

int n=s.nextInt();

RoundRobin obj=new RoundRobin(n);

obj.get();

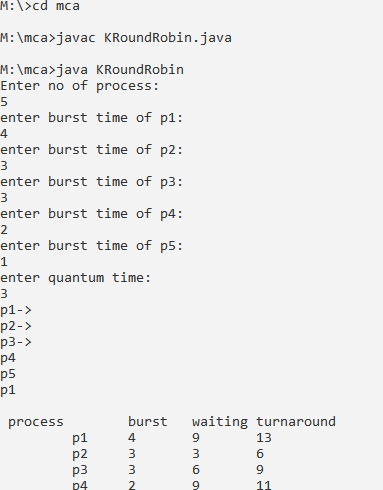
obj.round();

obj.display();

}

}

**OUTPUT:**





BANKERS ALGORITHM

import java.util.\*;

public class Bankers

{

private int need[][],allocate[][],max[][],avail[][],nr,np;

private void input()

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter number of processses and resources");

np=sc.nextInt();

nr=sc.nextInt();

need=new int[np][nr];

max=new int[np][nr];

allocate=new int[np][nr];

avail=new int[1][nr];

System.out.println("Enter allocation matrix-->");

for(int i=0;i<np;i++)

for(int j=0;j<nr;j++)

allocate[i][j]=sc.nextInt();

System.out.println("Enter max matrix-->");

for(int i=0;i<np;i++)

for(int j=0;j<nr;j++)

max[i][j]=sc.nextInt();

System.out.println("Enter available matrix-->");

for(int j=0;j<nr;j++)

avail[0][j]=sc.nextInt();

sc.close();

}

private int[][] calc\_need()

{

for(int i=0;i<np;i++)

for(int j=0;j<nr;j++)

need[i][j]=max[i][j]-allocate[i][j];

return need;

}

private boolean check(int i)

{

for(int j=0;j<nr;j++)

if(avail[0][j]<need[i][j])

return false;

return true;

}

public void isSafe()

{

input();

calc\_need();

boolean done[]= new boolean[np];

int j=0;

while(j<np)

{

boolean allocated=false;

for(int i=0;i<np;i++)

if(!done[i]&&check(i))

{

for(int k=0;k<nr;k++)

avail[0][k]=avail[0][k]-need[i][k]+max[i][k];

System.out.println("allocated process:"+i);

allocated=done[i]=true;

j++;

}

if(!allocated)

break;

}

if(j==np)

System.out.println("\n safely allocated");

else

System.out.println(" all process cant be allocated safely");

}

public static void main(String arg[])

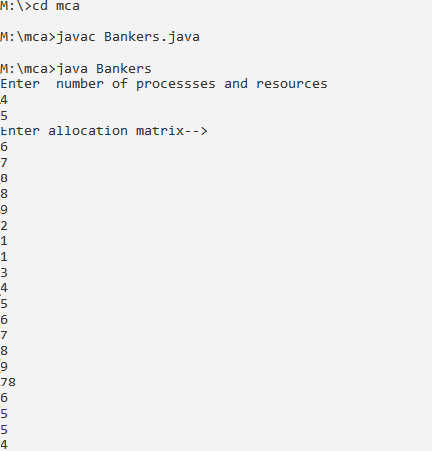
{

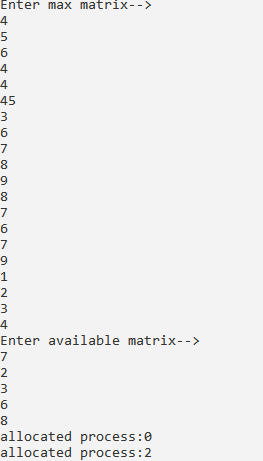
new Bankers().isSafe();

}

}

OUTPUT:





PAGE REPLACE (FIFO)

import java.io.\*;

class Replacepage

{

public static void main(String arg[]) throws IOException

{

BufferedReader obj=new BufferedReader(new InputStreamReader(System.in));

int f,page=0,ch,pgf=0,chn=0,n;

boolean flag;

int pages[];

System.out.println("Menu");

System.out.println("1.FIFO");

System.out.println("2.LRU");

System.out.println("3.LQR");

System.out.println("4.EXIT");

System.out.println("Enter ur choice");

ch=Integer.parseInt(obj.readLine());

switch(ch)

{

case 1:

int pt=0;

System.out.println("enter no of frames");

f=Integer.parseInt(obj.readLine());

int frame[]=new int[f];

for(int i=0;i<f;i++)

{

frame[i]=-1;

}

System.out.println("Enter no of pages");

n=Integer.parseInt(obj.readLine());

pages=new int[n];

System.out.println("Enter page no");

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

pages[j]=Integer.parseInt(obj.readLine());

do

{

int pg=0;

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

{

page=pages[pg];

flag=true;

for(int j=0;j<f;j++)

{

if(page==frame[j])

{

flag=false;

break;

}

}

if(flag)

{

frame[pt]=page;

pt++;

if(pt==f)

pt=0;

System.out.print("Frame:");

for(int j=0;j<f;j++)

System.out.print(frame[j]+" ");

System.out.println();

pgf++;

}

else

{

System.out.print("frame:");

for(int j=0 ;j<f;j++)

System.out.print(frame[j]+" ");

System.out.println();

}

chn++;

}

}

while (chn!=n);

System.out.println("page fault:"+pgf);

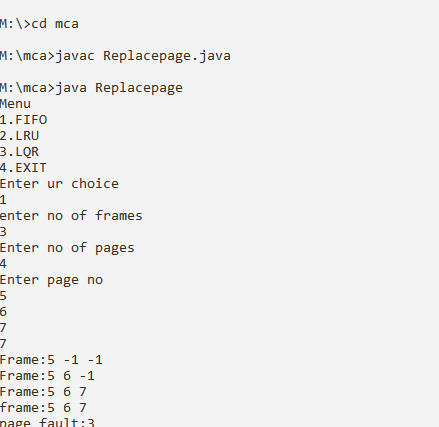
break;

}

}

}

OUTPUT:



DYNAMIC BINDING

import java.util.Scanner;

public class Dining

{

public static int tph,howhung,cho;

public static int[] philname = new int[20], status =new int[20],hu=new int[20];

public static void main(String[] args)

{

System.out.print("\nDINING PHILOSOPHER PROBLEM\n");

System.out.print("\nEnter the total no. of philosophers: ");

tph=STDIN\_SCANNER.nextInt();

for(int i=0; i<tph; i++)

{

philname[i]=i+1;

status[i]=1;

}

System.out.print("How many are hungry :");

howhung=STDIN\_SCANNER.nextInt();

if(howhung==tph)

{

System.out.print("\nAll are hungry..\n Dead lock stage will occur");

System.out.println("Exiting");

}

else

{

for(int i=0; i<howhung;i++)

{

System.out.print("Enter philosopher position"+ (i+ 1) + "p");

hu[i]=STDIN\_SCANNER.nextInt();

status[hu[i]]=2;

}

do

{

System.out.print("1.One can eat at a time\t2.Two can eat at a time\t3.Exit\nEnter your choice:");

cho=STDIN\_SCANNER.nextInt();

switch(cho)

{

case 1:

one();

break;

case 2:

two();

break;

case 3:

System.exit(0);

default:

System.out.print("I");

}

}

while(true);

}

}

public static int one()

{

int pos=0;

System.out.print("\nAllow one philosopher to eat at any time\n");

for(int i=0; i<howhung; i++, pos++)

{

System.out.print("\nPis granted to eat" + philname[hu[pos]]);

for(int x=pos; x< howhung; x++)

{

System.out.print("\nPis waiting" + philname[hu[x]] + " is waiting");

}

}

return 0;

}

public static int two()

{

int s=0,t,r;

System.out.print("\nAllow two philosophers to eat at same time\n");

for(int i = 0; i<howhung; i++)

{

for(int j=i+1; j<howhung;j++)

{

if(Math.abs(hu[i]-hu[j])>=1 && Math.abs(hu[i]-hu[j])!= 4)

{

System.out.print("\n\ncombination" + (s+1));

t= hu[i];

r= hu[j];

s++;

System.out.print("\nP and are granted to eat" + philname[hu[i]]+ "p" + philname[hu[j]] +"eat");

for(int x = 0; x < howhung; x++)

{

if(hu[x]!=t&& hu[x]!=r)

{

System.out.print("\nP is waiting" + philname[hu[x]] + "i");

}

}

}

}

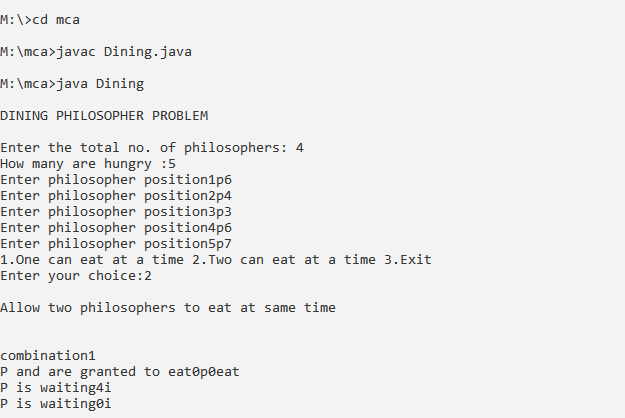
}

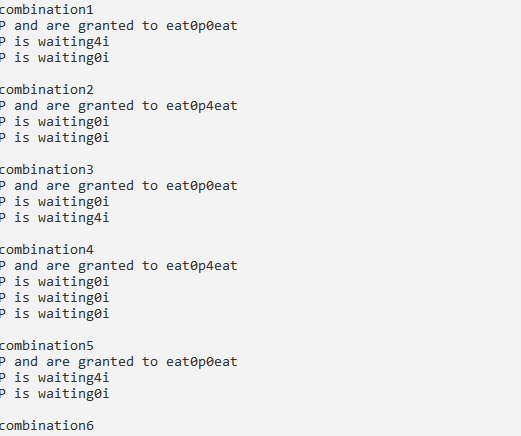
return 0;

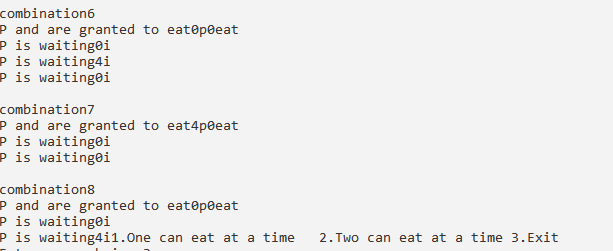
}

public final static Scanner STDIN\_SCANNER=new Scanner(System.in);

}







PRODUCER AND CONSUMER

import java.util.Scanner;

public class Producerconsumer

{

public static void main(String[] args)

{

int[] buffer=new int[10];

int bufsize,in,out,produce,consume,choice=0;

in=0;

out=0;

bufsize=10;

while(choice!=3)

{

System.out.print("\n1.produce \t 2.consume \t 3.exit");

System.out.print("\n enter your choice:");

choice=STDIN\_SCANNER.nextInt();

switch(choice)

{

case 1:

if((in+1)%bufsize==out)

{

System.out.print("\n buffer is full");

}else

{

System.out.print("\n enter the value:");

produce=STDIN\_SCANNER.nextInt();

buffer[in]=produce;

in=(in+1)%bufsize;

}

break;

case 2:

if(in==out)

{

System.out.print("\n buffer is empty");

}else

{

consume=buffer[out];

System.out.println("\n the consumed value is"+consume);

out=(out+1)%bufsize;

}

break;

}

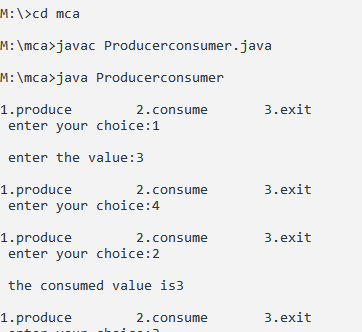
}

}

public final static Scanner STDIN\_SCANNER=new Scanner(System.in);

}

OUTPUT:



READ AND WRITE SEMAPHORES

import java.util.concurrent.Semaphore;

class RW

{

static Semaphore readLock=new Semaphore(1);

static Semaphore writeLock=new Semaphore(1);

static int readCount=0;

static class Read implements Runnable

{

public void run()

{

try{

readLock.acquire();

readCount++;

if(readCount==1)

{

writeLock.acquire();

}

readLock.release();

System.out.println("Thread"+Thread.currentThread().getName()+"is READING");

Thread.sleep(1500);

System.out.println("Thread"+Thread.currentThread().getName()+"has FINISHED READING");

readLock.acquire();

readCount--;

if(readCount==0)

{

writeLock.release();

}

readLock.release();

}

catch(InterruptedException e)

{

System.out.println(e.getMessage());

}

}

}

static class Write implements Runnable

{

public void run()

{

try

{

writeLock.acquire();

System.out.println("Thread"+Thread.currentThread().getName()+"is WRITING");

Thread.sleep(2500);

System.out.println("Thread"+Thread.currentThread().getName()+"has FINISHED WRITING");

writeLock.release();

}

catch(InterruptedException e)

{

System.out.println(e.getMessage());

}

}

}

public static void main(String args[])throws Exception{

Read read=new Read();

Write write=new Write();

Thread t1=new Thread(read);

t1.setName("thread1");

Thread t2=new Thread(read);

t2.setName("thread2");

Thread t3=new Thread(write);

t3.setName("thread3");

Thread t4=new Thread(write);

t4.setName("thread4");

t1.start();

t2.start();

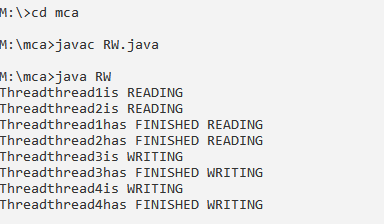
t3.start();

t4.start();

}

}

OUTPUT:



NAME AND VERSION OF OS

import java.util.\*;

public class GFG

{

private static final String nameOfOS=System.getProperty("Os.name");

private static final String versionOfOS=System.getProperty("sum.arch.data.model");

public static void main(String[] args)

{

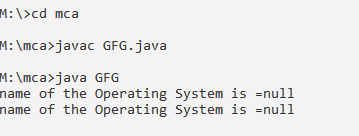
System.out.println("name of the Operating System is ="+nameOfOS);

System.out.println("name of the Operating System is ="+versionOfOS);

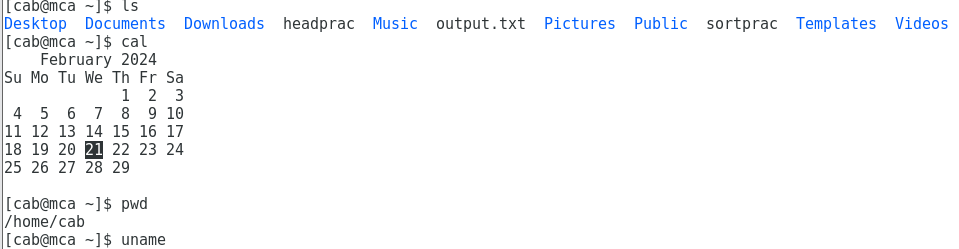
}

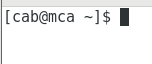
}

OUTPUT:

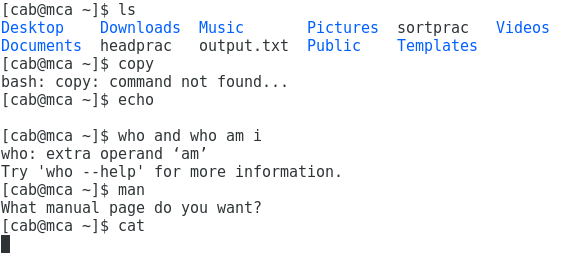


LINUX COMMANDS LS,CAL,PWD,UNAME,CLS

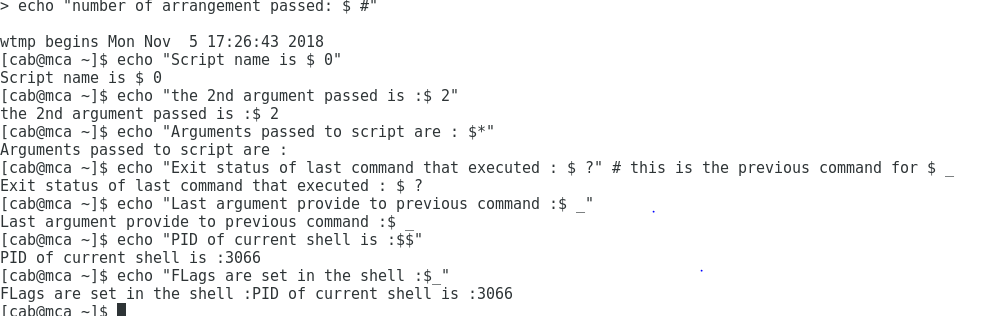




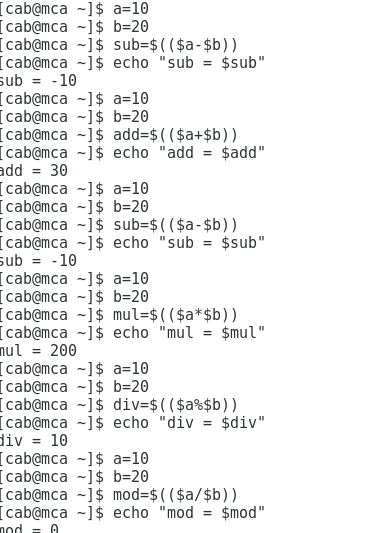
LINUX COMMANDS



LINUX COMMAND USAGE



LINUX ARITHMETIC OPERATOR DEMONSTRATION



BOOLEAN OPERATORS

