ASSIGNMENT-3

Q.1)

#include<unistd.h>

#include<stdlib.h>

#include<iostream>

#include<sys/types.h>

using namespace std;

int main()

{

pid\_t pid;

id=fork();

if(pid==0)

{

put<<"pid of child process----"<<getpid();

//child process

cout<<endl;

cout<<"ppid of child process----"<<getppid();

cout<<endl;

}

else if(pid>0)

{

cout<<"pid of parent process----"<<getpid();

//parent process

cout<<endl;

cout<<"ppid of parent process----"<<getppid();

cout<<endl;

}

else

{

cout<<"fork failed";

}

return 0;

}

\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*

pid of parent process----3021

ppid of parent process----3016

pid of child process----3022

ppid of child process—

Q.2)

#include<iostream>

#include<unistd.h>

#include<stdlib.h>

#include<sched.h>

using namespace std;

int main()

{

int a;

a=sched\_getscheduler(getpid());

switch(a)

{

case SCHED\_OTHER:cout<<"another scheduling poly"<<endl;

break;

case SCHED\_RR:cout<<"round robin scheduler"<<endl;

break;

case SCHED\_FIFO:cout<<"first in first out"<<endl;

break;

}

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*

another scheduling poly

\

Q.3)

#include<iostream>

#include<unistd.h>

#include<sched.h>

using namespace std;

int main()

{

int a;

a=sched\_setscheduler(getpid(),SCHED\_FIFO,0);

if(a==0)

{

cout<<"priority set";

}

else

{

cout<<"priority not set";

}

return 0;

}

\*\*\*Output\*\*\*\*

priority not set

Q.4)

#include<iostream>

#include<cstring>

using namespace std;

class process

{

public:

string name;

int arrTime;

int burstTime;

int respTime;

int compTime;

int waitTime;

int turnTime;

} ;

int main()

{

int i,noproc,avgwaitTime,avgturnTime,sum1 = 0,sum2 = 0;

cout<<"no of process=\n";

cin>>noproc;

process p[ noproc ];

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

{

cout<<"\n enter the process "<<i<<" name = ";

cin>>p[i].name;

cout<<"\n enter the process "<<i<<" arrival time = ";

cin>>p[i].arrTime;

cout<<"\n enter the process "<<i<<" burst time = ";

cin>>p[i].burstTime;

}

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

{

if( i== 0){

p[i].respTime = p[i].arrTime;

p[i].compTime = p[i].respTime + p[i].burstTime;

p[i].waitTime = p[i].respTime - p[i].arrTime;

p[i].turnTime = p[i].compTime - p[i].arrTime;

}

else

{

p[i].respTime = p[i-1].respTime + p[i-1].burstTime;

p[i].compTime = p[i].respTime + p[i].burstTime;

p[i].waitTime = p[i].respTime - p[i].arrTime;

p[i].turnTime = p[i].compTime - p[i].arrTime;

}

}

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

{

sum1= sum1 + p[i].waitTime;

sum2= sum2 + p[i].turnTime;

}

avgwaitTime = sum1/noproc;

avgturnTime = sum2/noproc;

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

{

cout<<"\n process no "<<i+1;

cout<<"\n name "<<p[i].name;

cout<<"\n arrival time "<<p[i].arrTime;

cout<<"\n burst time "<<p[i].burstTime;

cout<<"\n response time "<<p[i].respTime;

cout<<"\n complete time "<<p[i].compTime;

cout<<"\n waiting time "<<p[i].waitTime;

cout<<"\n turnaround time "<<p[i].turnTime;

cout<<endl;

}

cout<<"\n average waiting time = "<<avgwaitTime;

cout<<"\n average turnaround time ="<<avgturnTime;

return 0;

}

\*\*Output\*\*\*

no of process=

4

enter the process 0 name = P1

enter the process 0 arrival time = 12

enter the process 0 burst time = 5

enter the process 1 name = P2

enter the process 1 arrival time = 10

enter the process 1 burst time = 3

enter the process 2 name = p3

enter the process 2 arrival time = 15

enter the process 2 burst time = 7

enter the process 3 name = p4

enter the process 3 arrival time = 18

enter the process 3 burst time = 9

process no 1

name P1

arrival time 12

burst time 5

response time 12

complete time 17

waiting time 0

turnaround time 5

process no 2

name P2

arrival time 10

burst time 3

response time 17

complete time 20

waiting time 7

turnaround time 10

process no 3

name p3

arrival time 15

burst time 7

response time 20

complete time 27

waiting time 5

turnaround time 12

process no 4

name p4

arrival time 18

burst time 9

response time 27

complete time 36

waiting time 9

turnaround time 18

average waiting time = 5

average turnaround time =11