Q1

P1

|

P2

/ \

P3 P4

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<stdio.h>

int main(){

pid\_t pid;

printf("this is the pid id %d\n",getpid());

int first=fork();

wait();

if(first==0){

printf("this is the pid %d and parent ppid %d\n",getpid(),getppid());

int second=fork();

wait();

if(second==0){

printf("this is the pid %d and parent ppid %d\n",getpid(),getppid());

}

else{

int third=fork();

if(third==0)

printf("this is the pid %d and parent ppid %d\n",getpid(),getppid());

}

}

return 0;

}

Q2.

Execute than understand

#include<sys/types.h>

#include<unistd.h>

#include<stdio.h>

int main(){

pid\_t pid;

printf("This is pid %d\n",getpid());

int first=fork();

wait();

if(first==0){

printf("this is the pid %d and parent ppid %d\n",getpid(),getppid());

int third=fork();

wait();

if (third==0) {

printf("this is the pid %d and parent ppid %d\n",getpid(),getppid());

}

}

else{

int second=fork();

if(second==0){

printf("this is the pid %d and parent ppid %d\n",getpid(),getppid());

int four=fork();

wait();

if (four==0) {

printf("this is the pid %d and parent ppid %d\n",getpid(),getppid());

}

}

}

return 0;

}

Q3 producer consumer

#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("1.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");

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("producer produces the iteam %d\n",x);

mutex=signal(mutex);

}

void consumer()

{

mutex=wait(mutex);

full=wait(full);

empty=signal(empty);

x++;

printf("consumer consumes iteam %d\n",x-1);

x--;

mutex=signal(mutex);

}

Q4) reader writer problem

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<pthread.h>

#include<semaphore.h>

sem\_t w;

sem\_t r;

int a = 64;

int count = 0;

void \*reader()

{

  sem\_wait(&r);

   count++;

   if(count==1)

   {

   sem\_wait(&w);

   }

  sem\_post(&r);

   printf("\t\tReader is reading Content : %d\n",a);

   sleep(2);

   sem\_wait(&r);

   count--;

   if(count==0)

   {

   sem\_post(&w);

   }

   sem\_post(&r);

}

void \*writer()

{

   sem\_wait(&w);

   a=a/2;

   printf("\t\tWriter is writing Content : %d\n",a);

   sleep(1);

   sem\_post(&w);

}

int main()

{

printf("\n\n\t\tREADER WRITER program...\n\n\n");

   int i;

   sem\_init(&r,0,1);

   sem\_init(&w,0,1);

   pthread\_t r\_id[10],w\_id[10];

   for(i=1;i<6;i++)

   {

     pthread\_create(&w\_id[i],NULL,writer,NULL);

     pthread\_create(&r\_id[i],NULL,reader,NULL);

   }

   for(i=1;i<6;i++)

   {

     pthread\_join(w\_id[i],NULL);

     pthread\_join(r\_id[i],NULL);

   }

   printf("\t\tFinal value of our variable(global)   : %d\n\n\n",a);

   printf("\t\tReader-Writer ends here...    :)  ");

}