Operating System All Practicals

Practical 5

Write a program to deomnstare mutual exclusion and critical section.

public class MutualCritical {

public static void main(String[] args) {

MutualCritical mc = new MutualCritical();

new Thread(()->{

mc.criticalSection();

}).start();

new Thread(()->{

mc.criticalSection();

}).start();

}

public synchronized void criticalSection() {

System.out.println("In critical section");

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

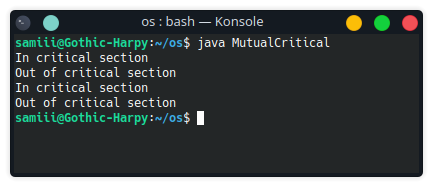
}

System.out.println("Out of critical section");

}

}

Output



Practical 6

Implement producer consumer program.

public class producerConsumer {

boolean produced = false;

public void produce() throws Exception {

if (produced) {

System.err.println("Producer: Already produced. Waiting for consumer to consume.");

return;

}

System.out.println("Producer: Producing...");

Thread.sleep(2000);

produced = true;

System.out.println("Producer: Produced.");

}

public void consume() throws Exception {

if (!produced) {

System.err.println("Consumer: Nothing to consume. Waiting for producer to produce.");

return;

}

System.out.println("Consumer: Consuming...");

Thread.sleep(2000);

produced = false;

System.out.println("Consumer: Consumed.");

}

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

producerConsumer pc = new producerConsumer();

pc.produce();

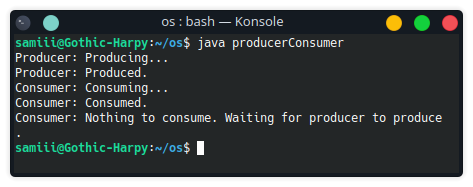
pc.consume();

pc.consume();

}

}

Output:



Practical 7

Implement program for reader and writer problem.

public class ReaderWriter {

boolean itemWrote, itemRead, isWriting, isReading;

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

ReaderWriter rw=new ReaderWriter();

rw.writer();

rw.reader();

}

public ReaderWriter() {

this.itemWrote = false;

this.itemRead = false;

}

public void writer() throws Exception {

if (this.isReading || this.isWriting) {

System.err.println("Writer: Cant write whiole reader or writer is active.");

return;

}

this.isWriting = true;

Thread.sleep(2000);

this.itemWrote = true;

this.isWriting = false;

System.out.println("Writer: Writing is done..");

}

public void reader() throws Exception {

if (this.isWriting) {

System.err.println("Reader: Cant read while writer is active.");

return;

}

if (!this.itemWrote) {

System.err.println("Reader: Nothing to read");

return;

}

this.isReading = false;

Thread.sleep(2000);

this.itemRead = true;

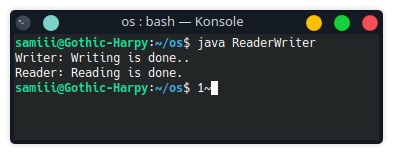
System.err.println("Reader: Reading is done.");

this.isReading = false;

}

}

Output:



Practical 8

Program to demonstrate CPU scheduling algorithm.

public class fcfs {

public static void main(String[] args) {

int[] process={0,1,2,3,4,5,6,7,8,9};

int[] arrival={0,1,2,3,4,5,6,7,8,9};

int[] burst={1,2,3,4,5,6,7,8,9,10};

int[] completion=new int[10];

int[] waiting=new int[10];

int[] turnaround=new int[10];

int[] temp=new int[10];

int total=0;

int n=10;

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

temp[i]=burst[i];

}

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

total+=burst[i];

completion[i]=total;

}

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

turnaround[i]=completion[i]-arrival[i];

waiting[i]=turnaround[i]-temp[i];

}

System.out.println("Process|Arrival|Burst|Completion|Turnaround|Waiting");

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

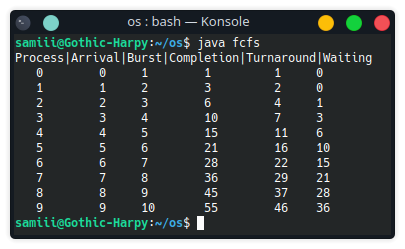
System.out.println(" "+process[i]+"\t "+arrival[i]+"\t "+temp[i]+"\t "+completion[i]+"\t "+turnaround[i]+"\t "+waiting[i]);

}

}

}

Output.java



Practical 9

Implement program to demonstrate deadlock detection.

public class deadlockDetection {

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

//Boolean.TRUE means any resource 1

//Boolean.FALSE means any resource 2

Thread thread1 = new Thread(() -> {

synchronized (Boolean.TRUE) {

System.out.println("Thread 1 acquired resource 1");

synchronized (Boolean.FALSE) {

System.out.println("Thread 1 acquired resource 2");

}

}

});

Thread thread2 = new Thread(() -> {

synchronized (Boolean.FALSE) {

System.out.println("Thread 2 acquired resource 2");

synchronized (Boolean.TRUE) {

System.out.println("Thread 2 acquired resource 1");

}

}

});

thread1.start();

thread2.start();

Thread.sleep(1000);

if(thread1.isAlive()&& thread2.isAlive()){

System.out.println("Deadlock detected");

}

else{

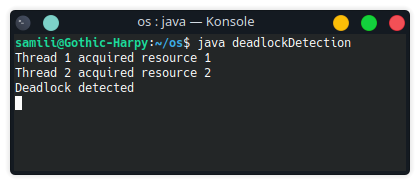
System.out.println("Deadlock not detected");

}

}

}

Output:



Practical 10

Imlpement program to demonstrate page replacement algorithm.

public class PageReplacement {

public static void main(String[] args) {

int[] pages = {1,2,3,4,1,2,5,1,2,3,4,5};

int[] frames = new int[3];

int pageFaults = 0;

int n = pages.length;

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

boolean pageFound = false;

for(int j=0; j<frames.length; j++){

if(frames[j] == pages[i]){

pageFound = true;

break;

}

}

if(!pageFound){

int minIndex = 0;

int minCount = Integer.MAX\_VALUE;

for(int j=0; j<frames.length; j++){

int count = 0;

for(int k=i-1; k>=0; k--){

if(frames[j] == pages[k]){

break;

}

count++;

}

if(count < minCount){

minIndex = j;

minCount = count;

}

}

frames[minIndex] = pages[i];

pageFaults++;

}

}

System.out.println("Page Faults: " + pageFaults);

}

}

Output:

