कम्प्यूटर विज्ञान एवं इंजीनियरिंग विभाग, मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान
भोपाल - 462003 (म.प्र.) भारत
Department of Computer Science & Engineering
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL – 462003 (M.P.) INDIA

ऑपरेटिंग सिस्टम प्रयोगशाला मैनुअल
Operating System Lab Manual
बी.टेक. V सेम (B. Tech. V Sem)

विषय सूची (TABLE OF CONTENTS)

--***---

प्रयोग नं. 1

EXPERIMENT:-1

a. वर्चुअल बॉक्स के साथ उबंटू की स्थापना

Installation of Ubuntu with virtual box

b. निम्नलिखित कमांड केआउटपुट के बारे में संक्षिप्त जानकारी लिखे

Brief about output of following commands

- **man**
- **ls**
- **bc**
- **cat**
- **cmp**
- **cp**
- **date**
- **df**
- **echo**
- **file**
- **mkdir**
- **mv**
- **pwd**
- **ps**
- **rm**
- **rmdir**
- **wc**

--***--

प्रयोग नं. 2

EXPERIMENT:-2

a. निर्देशिका संरचना- UNIX/LINUX

Directory Structure of UNIX / LINUX

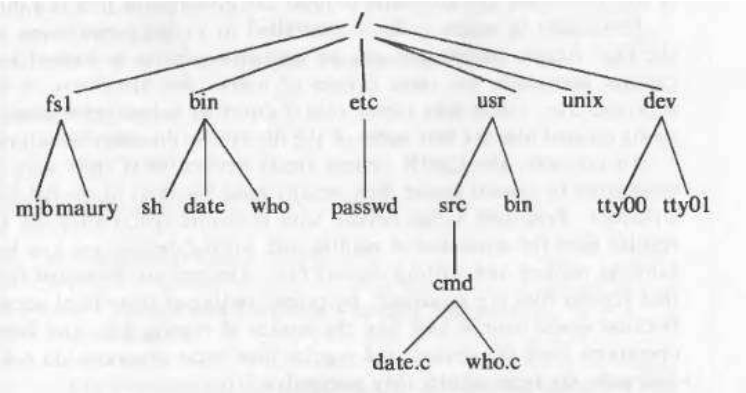


Figure 1 Sample Directory structure in UNIX/ LINUX [M.J. Bach "Design of UNIX O.S. ", PHI Learning]

"/" is root directory

a1. निम्नलिखित निर्देशिकाओं और उनकी सामग्री का संक्षिप्त विवरण लिखें:

Write about the brief details of following directories and their contents:

/bin

/dev

/etc

/home

/lib

/tmp

/usr

a2. लिनक्स में निर्देशिकाओं के पदानुक्रम को देखने के लिए ट्री कमांड का उपयोग करें।

Use tree command to visualize hierarchy of directories in Linux.

b. लिनक्स आंतरिक की खोज

Exploring Linux internals

b1. नीचे दिए गए फ़ोल्डरों/निर्देशिकाओं का अन्वेषण करें और इन निर्देशिकाओं के अंदर की सामग्री के बारे में अपने अवलोकन लिखें

Explore below folders/directories and write your observations about the content inside these directories

/lib/modules

/lib/modules/<*release of Linux version* >

/lib/modules/<*release of Linux version* >/kernel

/lib/modules/<*release of Linux version* >/kernel/drivers

/usr/src/linux-header< *release of Linux version* >/drivers

/usr/src/linux/Documentation/

निम्नलिखित प्रश्नों के उत्तर दीजिये

Answer following questions

b2. मॉड्यूल क्या है?

What is module?

b3.ko फ़ाइल क्या है?

What is .ko file?

b4.नीचे दिए गए आदेशों को निष्पादित करें रअपना अवलोकन लिखें और प्रत्येकआदेश के आउटपुट की व्याख्या भी करें

Execute below commands and write your observation also explain output of each command

$ modprobe –l

$ lsmod

$ cat /proc/cpuinfo

$ cat /proc/meminfo

$ cat /proc/partitions

$ lspci

---***---

प्रयोग नं. 3

EXPERIMENT:-3

a. लिनक्स (वीआईएम संपादक)

LINUX (VIM editor)

a1. VIM संपादक (जिसे **vi** संपादक भी कहा जाता है) केतरीकों को संक्षेप में समझाएं

Briefly explain modes of VIM editor (also known as vi editor)

- *Insert Mode*
- *Edit Mode*
- *Command Mode*

a2. VIM संपादक प्रत्येक मोड में पयोग की जाने वाली भिन्न कुंजियों और कमांडों को उनके उपयोग सहित सूचीबद्ध करें।

List out various keys and commands used in each mode of VIM editor with their use.

b. LINUX में फ़ाइल एक्सेस अनुमतिऔर स्वामित्व

File Access Permission and ownership in LINUX

b1. फ़ाइल एक्सेस अनुमतियों को व्यवस्थित करने के लिए chmod कमांड का अन्वेषण करें; इसके दो तरीके हैं, सापेक्ष मोड और निरपेक्ष मोड।

Explore *chmod* command for chaigning File Access Permissions; there are two ways, relative mode and absolute mode.

b2. फ़ाइल का स्वामी बदलने के लिए **chown** कमांड का अन्वेषण करें

Explore *chown* command for chaining owner of the file.

---***---

प्रयोग नं. 4

EXPERIMENT:-4

a. लिनक्स में सिस्टम कॉलऔर 'C' प्रोग्रामिंग का परिचय

Introduction to System calls and C programming in LINUX

a1.ओपन ( ), रीड ( ), और राइट ( ) सिस्टम कॉल का उनके इनपुट पैरामीटर और रिटर्न प्रकारों के साथ संक्षिप्त परिचय लिखें।

Write brief introduction of open ( ), read ( ), and write ( ) system calls with their input parameters and return types.

a2.लाइब्रेरीज़ पर संक्षिप्त नोट्स लिखें जिनका उपयोग आमतौर पर तब किया जाता है जब हम विभिन्न सिस्टम कॉल का उपयोग करना चाहते हैं।

Write brief notes on libraries that are generally used when we want to use different system calls.

b. C प्रोग्राम में सिस्टम कॉल का उपयोग करना

Using system calls in C programs

b1. एक फ़ाइल से दूसरी फ़ाइल में बाइट की संख्या कॉपी करने के लिए Open( ), Read( ) और Write( ) सिस्टम कॉल का उपयोग करके एक प्रोग्राम लिखें । प्रोग्राम इंटरैक्टिव होना चाहिएऔर उपयोगकर्ता से <मौजूदा फ़ाइल का नाम>, <नई फ़ाइल का नाम> और <कॉपी किए जाने वाले बाइट की संख्या>के रूप में इनपुट लेना चाहिए। उपरोक्त कार्यक्रम के निष्पादन के दौरान होने वाली सभी स्थितियों का ध्यान रखें। संकेत: लाइब्रेरीज़ जिनका आप उपयोग कर सकते हैं या प्रोग्रामों के ऊपर उपयोग किया जा सकता है

Write a program using open ( ), read ( ) and write ( ) system calls for copying number of byte from one file to another file. The program must be interactive and must take input from the user as <name of existing file>, <name of new file> and <number of byte to be copied>.

Take care of all the conditions that may happen during execution of above program.

Hint: libraries you may use or to be used above programs

#include <sys/types.h> , #include <studio.h>, #include <unistd.h> , #include<stdio.h>, #include<fcntl.h>

---***---

प्रयोग नं. 5

EXPERIMENT:-5

a. लिनक्स में प्रोसेस संबंधी सिस्टम कॉल का परिचय

Introduction to process related system calls in LINUX

a1. फोर्क ( ), एग्जीक्यूटिव ( ), और वेट ( ) सिस्टम कॉल का उनके इनपुट पैरामीटर और रिटर्न प्रकार के साथ संक्षिप्त परिचय लिखें।

Write brief introduction of fork ( ), exec ( ), and wait ( ) system calls with their input parameters and return types.

a2. निष्पादन ( ) सिस्टम कॉल के विभिन्न संस्करणों को उनके संक्षिप्त विवरण के साथ सूचीबद्ध करें।

List out the different versions of exec ( ) system call with their very brief details.

b. नई प्रक्रिया बनाने के लिए फोर्क का उपयोग करना

Using fork to create new process

b1. इनपुट के रूप में उपयोग कर्ता से पूर्णांक लेने के लिए एक प्रोग्राम लिखें और Fork( ) सिस्टम कॉल के साथ समान संख्या में चाइल्ड प्रोसेस उत्पन्न करें, मूल पीआईडी के साथ उनकी पीआईडी भी प्रिंट करें।

Write a program to take integer from user as input and generate same number of child process with fork ( ) system call, also print their PID along with parent PID.

b2. मूल प्रक्रिया चाइल्ड प्रक्रिया समाप्त होने की प्रतीक्षा कर रही है, यह प्रदर्शित करने के लिए एक प्रोग्राम लिखें।

Write a program to demonstrate that a parent process is waiting for child process to end.

---***---

प्रयोग नं. 6

EXPERIMENT:-6

a. सीपीयू शेड्यूलिंग एल्गोरिदम, एफसीएफएस, एसजेएफ, एसआरटीएफ, आरआर, गैर-प्रीमेप्टिव प्राथमिकता और प्रीमेप्टिव प्राथमिकता 'C' में सामान्य कोड लिखें। प्रोग्राम में पहले उपयोगकर्ता क्रम संख्या, आगमन समय, विस्फोट समय और प्राथमिकता जैसे अनुक्रम में डेटा इनपुट करेगा और फिर प्रत्येक प्रक्रिया और उनके औसत के लिए टीएटी, डब्ल्यूटी और आरटी प्राप्त करने के लिए एल्गोरिदम में से किसी एक का चयन करेगा। (यदि आरआर है) उपयोगकर्ता द्वारा चयनित तो इनपुट के रूप में समय की मात्रा भी लें)।

Write generic code in C of CPU Scheduling Algorithms, FCFS, SJF, SRTF, RR, non-Preemptive Priority, and Preemptive Priority. In the program first user will input data in sequence like, process no., arrival time, burst time, and priority and then select any one of the algorithm to get TAT, WT and RT for each process and their averages.(If RR is selected by user then also take time quantum as input).

b. दिए गए डेटा के लिए आरआर एल्गोरिदम के लिए 'C' में सामान्य सिमुलेशन कोड लिखें और विभिन्न समय क्वांटम मूल्यों का विश्लेषण करें और टी ए टी, डब्ल्यू टी और आर टी जैसे पैरामीटर के लिए समय क्वांटम का इष्टतम मूल्य पता लगाएं।

Write generic simulation code in C for RR algorithm for given data and perform the analysis of different time quantum values and find out the optimal value of time quantum for the parameter like, TAT, WT, and RT.

Process	Arrival Time	Burst Time
P1	0	10
P2	1	4
P3	3	6
P4	4	7
P5	5	2

---***---

प्रयोग नं. 7
EXPERIMENT:-7

a. बहुस्तरीय फीडबैक कतार का अनुकरण करने के लिए एक 'C' प्रोग्राम लिखें। 3 कतारें बनाएं, पहली कतार में आरआर2, दूसरी कतार में आरआर4 और तीसरी कतार में एफसीएफएस। एक सीपीयू घड़ी बनाएं जिसके अनुसार प्रत्येक कतार को सीपीयू समय दिया जाए। सीपीयू समय को तीन कतारों में विभाजित किया गया है, पहली कतार के लिए 8 समय क्वांटम, दूसरी कतार के लिए 16 समय क्वांटम और तीसरी कतार के लिए 32 समय क्वांटम। (कार्यान्वयन संरचना की लिंक सूची कीअवधारणा का उपयोग करके अनुकरण है, जिसमें पीआईडी, विस्फोट समय, शेष विस्फोट समय इत्यादि जैसे कई क्षेत्र हैं) अपना कोड जांचने के लिए नीचे दिए गए डेटा पर विचार करें:

Write a C program to simulate multilevel feedback queue. Create 3 queues, with RR2 in first queue, with RR4, in second queue and FCFS in third queue. Create one CPU clock according to which CPU time is given to each queue. CPU time is divided among three queues, 8 time quantum for first queue, 16 time quantum for second queue and 32 time quantum for third queue. (Implement is simulation by using concept of link list of structure, having several fields like PID, Burst time, Remaining Burst time, etc.) Consider below data to check your code:

Process No.	Burst time
P1	4
P2	2
P3	6
P4	15

आप मान सकते हैं कि सिमुलेशन की शुरुआत में सभी प्रक्रियाएं पहली कतार में हैं। प्रत्येक प्रक्रिया के लिए टीएटी, आरटी और डब्ल्यूटी का मूल्य और पूरे सिस्टम का उनका औसत मूल्य भी प्रिंट करें।

You may consider all processes are in first queue in the beginning of simulation. Also print value of TAT, RT and WT for each process and their average values of entire system.

---***---

प्रयोग नं. 8

EXPERIMENT:-8

a. प्रोसेस से संबंधित लिनक्स कमांड

Process Related Linux commands

a1. लिनक्समें "bg" और "fg" कमांड के उपयोग का प्रदर्शन करें

Demonstrate the use of “bg” and “fg” command in Linux.

a2. लिनक्स में "top" कमांड के सभी क्षेत्रों का अन्वेषण करें

Explore all fields of “top” command in Linux

a3. लिनक्स में "nice" कमांड के उपयोग का प्रदर्शन करें

Demonstrate the use of “nice” command in Linux

a4. लिनक्समें "nohup" कमांड के उपयोग का प्रदर्शन करें

Demonstrate the use of “nohup” command in Linux

b. मुख्य मेमोरी संबंधित लिनक्स कमांड

Main Memory Related Linux commands

b1. अपने सिस्टम (लिनक्स) में पेज का आकार जानने का तरीका बताएं।

Describe a way to find out the page size in your system (Linux).

b2. लिनक्स में "free" कमांड के आउटपुट में सभी फ़ील्ड का अन्वेषण करें।

Explore all fields in the output of “free” command in Linux.

b3. proc/meminfoकी सामग्री का वर्णन करें

Describe the content of /proc/meminfo

b4.लिनक्स में "vmstat" कमांड के सभी क्षेत्रों का अन्वेषण करें

Explore all fields of “vmstat” command in Linux

---***---

प्रयोग नं. 9

EXPERIMENT:-9

a. तार्किक पते (उदाहरण के लिए, 215 बिट्स) को भौतिक पते (उदाहरण के लिए, 212 बिट्स) में अनुवाद करने के लिए एक प्रोग्राम लिखें। इसके लिए, पृष्ठ आकार (उदाहरण के लिए, 27 बिट्स) तय करें, फिर एक नमूना पृष्ठ तालिका बनाएं और प्रविष्टियों को मैन्युअल रूप से भरें। कुछ यादृच्छिक तार्किक पते उत्पन्न करें और उन्हें पृष्ठ तालिका में प्रविष्टियों के अनुसार भौतिक पते पर मैप करें। पृष्ठ तालिका प्रविष्टियों के साथ, आउटपुट के रूप में पूर्णांक और बाइनरी दोनों रूपों में उनके मैप किए गए भौतिक पते के साथ तार्किक पता प्रिंट करें।

Write a program to translate a logical address (for example, 215 bits) to a physical address (for example, 212 bits). For this, fix the page size (for example, 27 bits), then create a sample page table and fill entries manually. Generate some random logical addresses and map them to physical addresses as per the entries in the page table. Print the logical address with their mapped physical address both in integer and binary form as output, along with page table entries.

b. पेजिंग योजना को कार्यान्वित करें, उपयोगकर्ता से इनपुट के रूप में प्रक्रियाओं की संख्या और उनका आकार लें, और इनपुट के रूप में रैम का आकार और पेज आकार भी लें। अब प्रत्येक प्रक्रिया से यादृच्छिक अनुक्रम में तार्किक पतों की पीढ़ी शुरू करें और फ्रेम की मुफ्त सूची के अनुसार उन्हें मुफ्त फ्रेम आवंटित करें। एक बार जब सभी फ्रेम पेजों से लोड हो जाएं, तो प्रोग्राम को रोकें और प्रत्येक प्रक्रिया के पेज टेबल की स्थिति, फ्रेम नंबर और उनमें लोड किए गए प्रोसेस नंबर और पेज नंबर के साथ प्रिंट करें। इसके लिए, आपको कुछ संरचनाएं और फ़ंक्शंस बनाने की ज़रूरत है जैसे पेज टेबल, उनकी स्थिति के साथ फ़्रेम की एक सूची, एक फ़्रेम एलोकेटर, आदि।

Implement Paging Scheme, take number of process with their size as input from user, also take size of RAM and page size as input. Now randomly start generation of logical address in random sequence from each process and allocate them free frames as per the free list of frames. Once all the frames are loaded with pages, stop the program and print the status of page tables of each process and also print frame numbers along with their process number and page number loaded in it.

For this you need to create some structures and function like page table, list of frames with their status, frame allocator, etc.

---***---

प्रयोग नं. 10

EXPERIMENT:-10

a. निश्चित विभाजन (मल्टीप्रोग्रामिंग की डिग्री निश्चित है): इन दोनों पर विचार करके उपयोगकर्ता से इनपुट के रूपमें ब्लॉक (या, मेमोरी विभाजन) की संख्या लें -

Implement the memory placement schemes with the fixed and variable partitioning methods as per the following:-

Fixed Partitioning (Degree of multiprogramming is fixed): Take the number of blocks (or, memory partitions) as input from user by considering both these –

A. समान ब्लॉक आकार

Equal block sizes

B. असमान ब्लॉक आकार

Unequal block sizes

प्रक्रियाओं को उनके आकार के साथ उपयोगकर्ता से इनपुट के रूप में भी लिया जाता है। अब, प्रोग्राम को सभी मेमोरी ब्लॉक को खाली के रूप में प्रारंभ करना चाहिए।और फिर, "फर्स्टफिट" और "बेस्टफिट" तरीकों को लागू करके, प्रक्रियाओं को मेमोरी ब्लॉक में समायोजित करें।आपके प्रोग्राम को समायोजित प्रक्रियाओं के लिए मेमोरी ब्लॉक स्थिति प्रिंट करनी चाहिए। इसके अलावा, प्रत्येक ब्लॉक के "आंतरिक विखंडन" की गणना के साथ-साथ "बाह्यविखंडन" (यदि कोई हो) और "कुल आंतरिक विखंडन" की गणना के लिए एक अलग प्रक्रिया लिखें।

The processes are also taken as input from the user with their sizes. Now, the program should initialize all the memory block as empty. And then by applying the “First Fit” and “Best Fit” methods, accommodate the processes in the memory blocks. Your program should print the memory blocks status with accommodated processes. Also write the separate procedure to calculate the “Internal Fragmentation” of each block along with the computation of “External Fragmentation” (if there exist) and the “Total Internal Fragmentation”.

Example: CASE – 1

Size of Main Memory = 1024 KB
Total Number of Blocks: 4
Equal Size Blocks – {256 KB, 256 KB, 256 KB, 256 KB}
Processes Size – {100 KB, 150 KB, 200 KB, 250 KB}

Example: CASE – 2

Size of Main Memory = 1024 KB
Total Number of Blocks: 4
Unequal Size Blocks – {128 KB, 128 KB, 256 KB, 512 KB}
Processes Size – {100 KB, 150 KB, 200 KB, 250 KB}

---***---

प्रयोग नं. 11

EXPERIMENT:-11

a. निम्नलिखित के अनुसार परिवर्तनीय विभाजन विधियों के साथ मेमोरी प्लेसमेंट योजनाओं को कार्यान्वित करें: -

परिवर्तनीय विभाजन (मल्टीप्रोग्रामिंग की डिग्री परिवर्तनीय है): मेमोरी ब्लॉक को इनपुट में बड़े आकार के छेद के रूप में लें (या, एकल मेमोरी विभाजन)। प्रक्रियाओं को उनके आकार के साथ उपयोगकर्ता से इनपुट के रूप में भी लिया जाता है। अब, प्रोग्राम को मेमोरी ब्लॉक को खाली के रूप में प्रारंभ करना चाहिए। और फिर "वर्स्ट फिट" विधि को लागू करके, प्रक्रियाओं को मेमोरी ब्लॉक में समायोजित करें। समय पर आदेशित घटनाओं पर विचार करके, आपके प्रोग्राम को प्रत्येक प्रक्रिया द्वारा व्याप्त मेमोरी आकार और उपलब्ध मेमोरी की वर्तमान स्थिति को प्रिंट करना चाहिए। "बाह्य विखंडन" की गणना के लिए एक अलग प्रक्रिया भी लिखें।

Implement the memory placement schemes with the variable partitioning methods as per the following:-

Variable Partitioning (Degree of multiprogramming is Variable): Take the memory block as large size hole in the input (or, single memory partition). The processes are also taken as input from the user with their sizes. Now, the program should initialize the memory block as empty. And then by applying the “Worst Fit” method, accommodate the processes in the memory blocks. By considering the timely ordered events, your program should print the memory size occupied by each process and the current status of available memory. Also write a separate procedure to calculate “External Fragmentation”.

Example: CASE – 1

Size of Main Memory = 1024 KB
Total Number of Blocks: 1
Time Events – All processes are arrived at same time T.
T (Process Arrival): P1 (100 KB), P2 (200 KB), P3 (300 KB), P4 (300 KB), P5 (300 KB)

Example: CASE – 2

Size of Main Memory = 1024 KB
Total Number of Blocks: 1
Time Events – The process arrivals are different.
T1 (Process Arrived): P1 (100 KB)
T2 (Process Arrived): P2 (100 KB)
T3 (Process Arrived): P3 (200 KB)
T4 (***Process Leaves***): P2 (100 KB)
T5 (Process Arrival): P4 (200 KB)
T6 (Process Arrival): P5 (200 KB)
T7 (***Process Leaves***): P4 (200 KB)
T8 (Process Arrival): P6 (300 KB)

---***---

प्रयोग नं. 12
EXPERIMENT:-12

a. नीचे उल्लिखित अपेक्षित डेटा संरचनाओं को लेकर गतिरोध से बचने के लिए बैंकर के एल्गोरिदम का अनुकरण करने के लिए एक प्रोग्राम लिखें, जहां M = प्रक्रियाओं की संख्या और N = संसाधन प्रकारों की संख्या। "संसाधन आवंटन एल्गोरिथम" और "सुरक्षा एल्गोरिथम" के लिए अलग-अलग प्रक्रियाएं लागू करें। आपके प्रोग्राम को डेटा संरचनाओं के लिए आवश्यक तात्कालिक इनपुट मान लेना चाहिए, और यह सिस्टम की आउटपुट स्थिति को "सुरक्षित स्थिति" या "असुरक्षित स्थिति" के रूप में उत्पन्न करने की उम्मीद है।

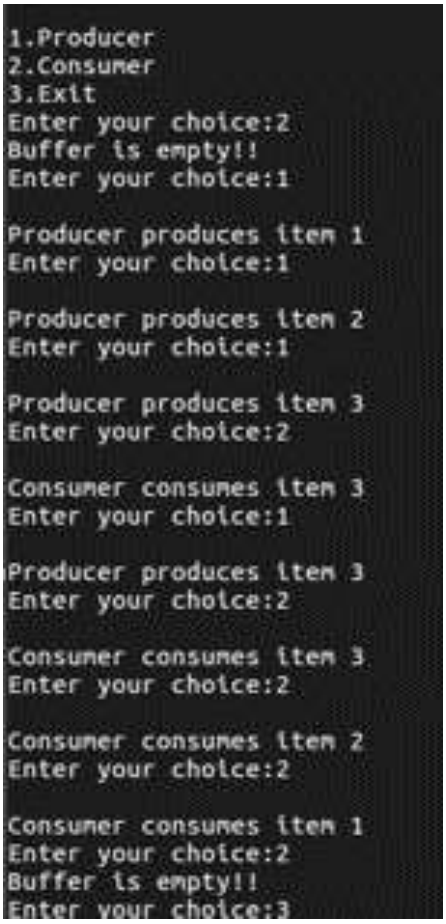
Write a program to simulate the **Banker's Algorithm** for deadlock avoidance by taking the requisite data structures mentioned below, where $\boldsymbol{M}$ = no. of processes and $\boldsymbol{N}$ = no. of resource types. Implement the separate procedures for the "**Resource Allocation Algorithm**" and the "**Safety Algorithm**". Your program must take the instantial input values required for the data structures, and it is expected to produce the output state of the system as either "**Safe State**" or "**Unsafe State**".

- Total Available Resources Matrix: $[\boldsymbol{TOTAL}]_{\boldsymbol{1\times N}}$
- Maximum Allocation Matrix / Claim Matrix**:**$[\boldsymbol{MAXIMUM}]_{\boldsymbol{M\times N}}$
- Allocated Matrix / Current Allocation Matrix**:**$[\boldsymbol{CURRENT}]_{\boldsymbol{M\times N}}$
- Need Matrix**:**$[\boldsymbol{NEED}]_{\boldsymbol{M\times N}} = [\boldsymbol{MAXIMUM}]_{\boldsymbol{M\times N}} - [\boldsymbol{CURRENT}]_{\boldsymbol{M\times N}}$
- Currently Available Resources Matrix: $[\boldsymbol{AVAILABLE}]_{\boldsymbol{1\times N}}$

b. सुप्रसिद्ध शास्त्रीय तुल्य कालन समस्या "निर्माता-उपभोक्ता" है। वे दोनों एक निश्चित आकार का बफ़र साझा करते हैं (क्यू के रूप में उपयोग किया जाता है)।निर्माता डेटा उत्पन्न कर सकता है और इसे बफर में रख सकता है ।उपभोक्ता को बफ़र से डेटा का उपभोग करना चाहिए (एक समय में एक)। यदि बफ़र भरा हुआ है तो निर्माता डेटा नहीं रख सकता है, और बफ़र खाली होने पर उपभोक्ता डेटा का उपभोग नहीं कर सकता है। "निर्माता-उपभोक्ता" समस्या का अनुकरण करें क्योंकि आपका निष्पादन और आउटपुट इस प्रकार होगा:

The well-known classical synchronization problem is "**Producer – Consumer**". They both share a fixed-size buffer (used as a queue). The producer can generate the data and keep it in

the buffer. The consumer is supposed to consume the data from the buffer (one at a time). Producer cannot place the data if the buffer is full, and Consumer cannot consume the data when the buffer is empty. Simulate the "**Producer – Consumer**" problem as your execution and output will be as follows:



---***----

प्रयोग नं. 13

EXPERIMENT:-13

a. सेमाफोर का कार्यान्वयन: एक प्रोग्राम लिखें जो दर्शाता है कि सेमाफोर का उपयोग करके दो प्रक्रियाएं एक चर को कैसे साझा कर सकती हैं। बाइनरी सेमाफोर (पारस्परिक बहिष्करण प्राप्त करने के लिए) और काउंटिंग सेमाफोर (प्रक्रियाओं के बीच सिंक्रनाइज़ेशन के लिए) का अनुकरण करने के लिए उपलब्ध संसाधनों की संख्या के आधार पर स्वतंत्र उदाहरण बनाएं।

Implementation of Semaphore: Write a program that demonstrates how the two processes can share a variable using Semaphore. Create independent instances on the basis of number of available resources to simulate the Binary Semaphore (to achieve the mutual exclusion) and Counting Semaphore (for the synchronization among the processes).

b. संसाधन आवंटन ग्राफ (आरएजी) लागू करें: निम्नलिखित इनपुट धारणाओं को लेकर सिस्टम की स्थिति का प्रतिनिधित्व करने के लिए आरएजी का अनुकरण करें:

Implement the Resource Allocation Graph (RAG): Simulate the RAG to represent the state of the system by taking the following input assumptions:

- प्रक्रियाओं की संख्या (P1, ..., Pm)
  Number of processes (P1, ..., Pm)
- एकल उदाहरण प्रकार के संसाधनों की संख्या (R1, ……., Rn)
  Number of single instance type resources (R1, ……., Rn)
- प्रत्येक प्रक्रिया के लिए आवंटित संसाधनों की संख्या (असाइनमेंट एज को अनुकरण करने के लिए)
  Number of resources allocated to each process (to simulate the assignment edge)
- प्रत्येक प्रक्रिया द्वारा अभी भी अनुरोध किए जा रहे संसाधनों की संख्या (अनुरोध किनारे का अनुकरण करने के लिए)
  Number of resources, still requesting by each process (to simulate the requesting edge)

गतिरोध की घोषणा के लिए RAG में चक्र का पता लगाने के लिए एक अलग प्रक्रिया लिखें। और यदि कोई चक्र नहीं है, तो आपके प्रोग्राम को सिस्टम की स्थिति को गतिरोध मुक्त के रूप में आउटपुट करना चाहिए।

Write a separate procedure to detect the cycle in RAG to declare for the deadlock. And if there is no cycle, then your program should output the state of the system as deadlock free.

---***---

References:

- Operating system concepts" Abraham Silberschatz and Peter B Galvin
- The Design of the UNIX Operating System by Maurice J. Bach
- Operating system a concept based Approach" D.M Dhamdhere

प्रयोगशाला समन्वयक (Lab Coordinators)
डॉ. मनीष पांडे **(**Dr. Manish Pandey**)**
डॉ.मितुल कुमार अहिरवाल (Dr. Mitul Kumar Ahirwal)
डॉ. राजेश वाधवानी (Dr. Rajesh Wadhvani)
डॉ.वासुदेवदेहलवार(Vasudev Dehalwar)
डॉ. धीरेंद्र प्रताप सिंह **(**Dr. Dhirendra Pratap Singh)

शिक्षणसहायक(Teaching Assistant)
सुशील चतुवेर्दी (SushilChaturvedi**)**
खुशबू सिंह(Khushboo Singh**)**
ललित कुमार (Lalit Kumar**)**
साधना बिजरोठिया(Sadhana Bijrothiya**)**
मुक्तेश गुप्ता (Muktesh Gupta**)**
राहुल सिंह (Rahul Singh**)**
बरखा सोनी (Barkha Soni**)**

---*---**