



Date _____

Date _____

Page No.: 22

Date.:

Tutorial - 4

Q1 Write an algorithm of producer consumer problem for infinite buffer. Explain with the help of tracing.

⇒ semaphore $n=0, s=1$

void producer()

{
while (true)

produce();

semWait (s);

append();

semSignal (s);

semSignal (n);

void consumer()

{

while (true)

{

semWait (n);

semWait (s);

take();

semSignal (s);

consume();

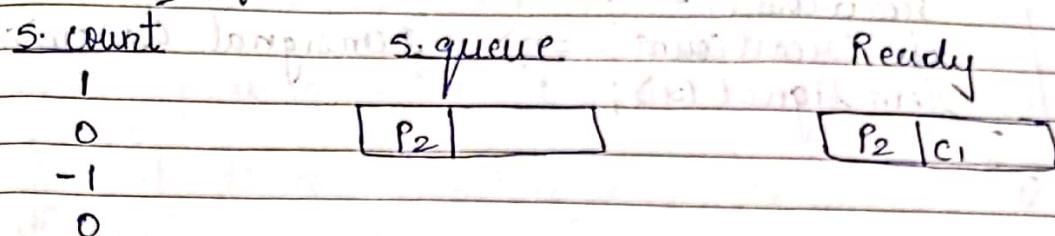
} }

void main () {

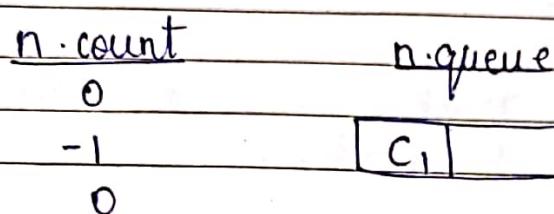
parbegin (producer, consumer)

}

Tracing → finding semaphore values



buffer is
full
empty



Q2 Write an algorithm of reader-writer problem having writers have priority. Explain this with help of tracing:

⇒ int readcount, writecount, rsem, wsem;

void reader ()

{

while (true) {

 semWait (rsem);

 semWait (wsem);

 semWait (rsem);

 if (readCount == 1)

 semWait (wsem);

 semSignal (wsem);

 semSignal (rsem);

 READUNIT ();



```
semWait(x);
readCount--;
if (readCount == 0) semSignal(wssem);
semSignal(x);
```

```
void writer()
```

```
{
```

```
while (true)
```

```
semWait(y);
```

```
writeCount++;
```

```
if (writeCount == 1)
```

```
semWait(wssem);
```

```
semSignal(y);
```

```
semWait(wssem);
```

```
WRITE UNIT();
```

```
semSignal(wssem);
```

```
semWait(y);
```

```
writeCount;
```

```
if (writeCount == 0) semSignal(wssem);
```

```
semSignal(y);
```

```
void main()
```

```
{
```

```
readCount = writeCount = 0;
```

```
parbegin(reader, writer);
```

Q3 Explain Message Passing & explain producer consumer algorithm using message passing.

⇒ Message passing refers to means of communication between:-

- different threads within a process.
- different processes running on same node.
- different processes running on different node.

Message has predefined structure and message passing uses two system call: Send & Receive

Send (destination, message)

Receive (source, message)

This is minimum set of operations needed for processes to engage in message passing. A process sends information in the form of message to another process designated by a destination. A process receives information by executing the receive primitive, indicating the source and the message.

```
const int capacity = /* buffering capacity */;  
null = /* empty message */;  
int i;
```

void producer()

message (msg);

while (true)

receive (may produce, msg);

pmsg = producer();

send (may consume, pmsg);

g

void consumer()

g

message (msg);

while (true)

receive (may consume, msg);

consume (msg);

send (may produce, null);

void main()

g

create mailbox (may produce);

create mailbox (may consume);

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

send (may produce, null);

parbegin (producer, consumer);

g

Q) Write difference between semaphore & Monitor

Semaphore

- 1) A variable used to control access to a common resource by multiple processes in a concurrent system such as a multitasking operating system.

Monitor

- 1) A synchronization construct that allows threads to have both mutual exclusion and ability to wait (block) for a certain condition to become true.

- 2) An integer variable.

- 2) An abstract datatype.

- 3) There is no condition variable concept.

- 3) Has condition variables.

- 4) When a process requires to access the semaphore, it performs wait() & performs signal() when releasing the resource.
- 4) A process uses procedures to access the shared variable in the monitor.

Tutorial 3



Page No.: 29
Date:

Q1 Fetch-And-Add (l, i) is an atomic read-modify-memory location l , increments it by the value i , and returns the old value of l . It is used in the pseudocode shown below to implement a busy-wait lock. l is an unsigned integer shared variable initialized to 0. The value of 0 corresponds to lock being available, while any non-zero value corresponds to the lock being not available.

Require lock (l):
while (Fetch-And-Add ($l, 1$))

$l = l + 1$; // busy-wait loop to wait

Release lock (l):
 $l = 0$;

This implementation is not safe, since if multiple processes enter the loop at the same time, they will both increment l .

- (A) Fails as l can overflow.
- (B) Fails as l can take on a non-zero value when the clock is actually available.
- (C) Works correctly but may starve some processes.
- (D) Works correctly without starvation.

Q2: Each process $P_i, i=0, 1, 2, 3, \dots, 9$ is coded as follows:-

repeat

$P(\text{mutex})$

{ Critical Section }

$V(\text{mutex})$

forever

Mutex

The code for P_{10} is identical except that it uses $V(\text{mutex})$ instead of $P(\text{mutex})$. What is the largest number of processes that can be inside the critical section at any moment (the mutex being initialized to 1)?

- a) 1 b) 2 c) 3 d) None of these

Q3: Two processes, P_1 and P_2 , need to access a critical section of code. Consider the following synchronization construct used by the processes:

Process P_1 :

while (true)

{ w1 = true;

while ($w_2 = \text{true}$);

Critical Section

$w_1 = \text{false};$

g

Remainder Section

Process P_2 :

while (true)

{ w2 = true;

while ($w_1 = \text{true}$);

Critical Section

$w_2 = \text{false};$

g

Remainder Section

Here, w_1 and w_2 are shared variables, which are initialized to false. Which one of the following statements is TRUE about the above construct?

- a) It does not ensure mutual exclusion.
- b) It does not ensure bounded waiting.
- c) It requires that processes enter the critical section in strict alternation.
- d) It does not prevent deadlocks, but ensures mutual exclusion.

Q4 The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as $s_0 = 1$, $s_1 = 0$, $s_2 = 0$.

Process P₀: while (true);
Process P₁: while (true);
Process P₂: while (true);
~~while (true); wait(s₁); wait(s₂);~~
~~wait(s₀); signal(s₀);~~
~~signal(s₁);~~
~~signal(s₂);~~

- Q3 How many times will P0 print '0' [2]?
- a) At least twice
 - b) Exactly twice
 - c) Exactly three
 - d) Exactly once

Q5 Consider the methods used by processes P_1 and P_2 for accessing their critical sections whenever needed as given below. The initial values of shared boolean variables s_1 and s_2 are randomly assigned.

Method used by P_1 :-

while ($s_1 = s_2$);

Critical section

$s_1 = s_2$;

Method used by P_2 :-

while ($s_1 \neq s_2$);

Critical section

$s_2 = \text{not}(s_1)$;

- Which of the following describes properties achieved?
- a) Mutual exclusion but not progress
 - b) Progress but not mutual exclusion
 - c) Neither mutual exclusion nor progress
 - d) Both mutual exclusion and progress

Tutorial-4

A computer has 8 tape drives, with n processes competing for them. Each process may need maximum three tape drives. For n which value of n is system deadlock free?

$$\underline{n = 3} \quad \underline{m_{\text{avc}} = 3}$$

System deadlock fuel = 3

$p_1 = 3$	3	$p_1 = 2$
$p_2 = 2$	3	$p_2 = 2$
$p_3 = 2$	2	$p_3 = 2$

Allocated Matrix

	X	Y	Z
P0	1	2	1
P1	2	0	1
P2	?	?	1

Request/Need Matrix

	x	y	z
P_0	1	0	3
P_1	0	1	2
P_2	1	2	0

Resource Vector(R)

Available Vector

i) P_{SI} will prevail

9	0	1
0	Φ	2
3	Φ	3

ii) PD will proceed

2 1 3
1 2 1
3 3 4

iii) P₂ will proceed

3 3 4

Hence P_2 will

proceed at last.

R1	R2	R3	R4
----	----	----	----

Q3 Resource (R)

Vector

At $t=0$

3	2	3	2
---	---	---	---

P₁(-) $\begin{matrix} 3 \\ 0 \end{matrix}$

P₂(-) $\begin{matrix} 1 \\ 0 \end{matrix}$

P₃(-) $\begin{matrix} 0 \\ 0 \end{matrix}$

At $t=1$

3	1	0	1
---	---	---	---

P₁(-) $\begin{matrix} 3 \\ 0 \end{matrix}$

P₂(-) $\begin{matrix} 1 \\ 0 \end{matrix}$

P₃(-) $\begin{matrix} 0 \\ 0 \end{matrix}$

At $t=2$

3	0	0	0
---	---	---	---

P₁(-) $\begin{matrix} 3 \\ 0 \end{matrix}$

P₂(-) $\begin{matrix} 0 \\ 0 \end{matrix}$

P₃(-) $\begin{matrix} 0 \\ 0 \end{matrix}$

At $t=3$

P₁(-) $\begin{matrix} 1 \\ 0 \end{matrix}$

P₂(-) $\begin{matrix} 0 \\ 0 \end{matrix}$

P₃(-) $\begin{matrix} 0 \\ 0 \end{matrix}$

(Requests 2 units of R₁, whereas we have only 1 unit of R₁ so this process will be blocked (P₁))

$$\cancel{At} \quad t = 4$$

P₂(-) 1

$$\cancel{Nt} - t = 5$$

$$P_1(+)$$

$$P_3(+)_2$$

$$3 \quad 1 \quad 0 \quad 0 \quad 1 - 1 \quad 10$$

2 Now Removal

block and

the requirement.)

1 8 P 1 P

1 1 1 301 1 1 617

18 19 20 21

Digitized by srujanika@gmail.com

1 1 8 2

~~1 2 0~~

110 2 0

$$x + t = 8$$

Handwritten notes showing three rows of numbers:

- P₁(+)**: Row of numbers 1, 0, 2, 0. Above the row, a wavy line starts at 1, goes down to 0, then up to 2, then down to 0. To the right of the row, it says "(P₁ blocked 2 units of R₁)".
- P₂(+)**: Row of numbers 1, 0, 2, 1. Above the row, a wavy line starts at 1, goes down to 0, then up to 2, then down to 1.
- P₃(-)**: Row of numbers 2, 0, 3, 1.

$$\cancel{At} \quad t = 9$$

$P_3(+)$	0	1	0	1	0	1	1
$P_1(-)$	0	1	0	1	0	1	1
$N + t = 10$	0	1	0	1	0	1	1
$P_1(+)$	1	2	0	2	1	1	1
	3	2	3	2			

so finally

A handwritten musical score on four-line staff paper. The score consists of four measures separated by vertical bar lines. The first measure contains a single note with a vertical stroke through it, followed by the number '3'. The second measure contains a single note with a vertical stroke through it, followed by the number '2'. The third measure contains a single note with a vertical stroke through it, followed by the number '3'. The fourth measure contains a single note with a vertical stroke through it, followed by the number '2'.

Allocated & Need

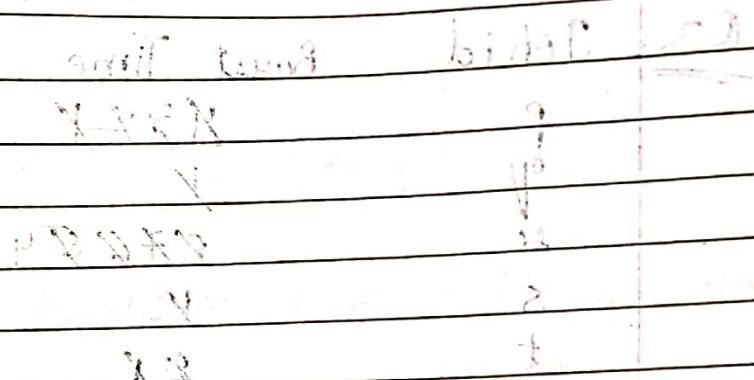
Who Max Has Room

A	3	1	2	1
B	2	1	1	1
C	4	2	2	1

B needs only 1 room
so B will be
granted and A
and C will be
blocked.

System 5 10 - 1

This is because no additional function is
used but user need memory which has been
allocated and user can't access it.

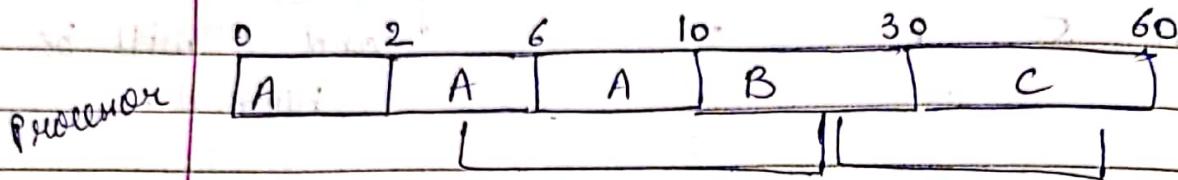


badham writes down

writing [x] [y] [z] [q] [t] [s] [l] [p] [f] [g] [m] [n]

Tutorial-6Q2

Process	Arrival Time	Service Time
A	0	10 8 X
B	2	20
C	6	30



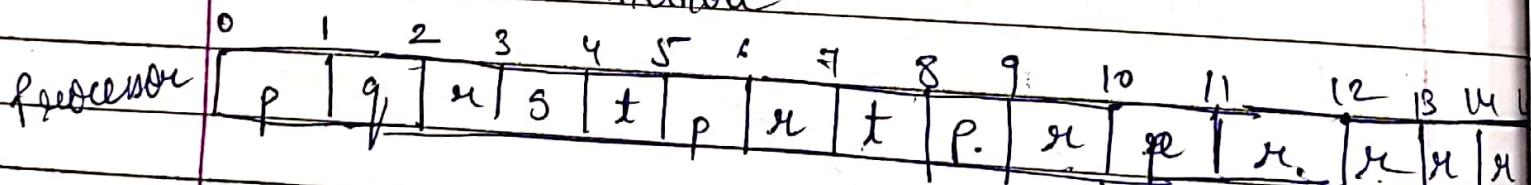
2 context switches are needed.

Context switching means how much time processes will change from processor.

Q3 Job id Burst Time

P	X 3 X X
Q	Y
U	8 X 6 X 4
S	Y
T	2 X

Round Robin method



Process Arrival time - Burst time

P1 0 84

P2 1 82

P3 2 81

P4 3 80

P5 4 79

P6 5 78

P7 6 77

P8 7 76

P9 8 75

P10 9 74

P11 10 73

P12 11 72

P13 12 71

P14 13 70

P15 14 69

P16 15 68

P17 16 67

P18 17 66

P19 18 65

P20 19 64

P21 20 63

P22 21 62

P23 22 61

P24 23 60

P25 24 59

P26 25 58

P27 26 57

P28 27 56

P29 28 55

P30 29 54

P31 30 53

P32 31 52

P33 32 51

P34 33 50

P35 34 49

P36 35 48

P37 36 47

P38 37 46

P39 38 45

P40 39 44

P41 40 43

P42 41 42

P43 42 41

P44 43 40

P45 44 39

P46 45 38

P47 46 37

P48 47 36

P49 48 35

P50 49 34

P51 50 33

P52 51 32

P53 52 31

P54 53 30

P55 54 29

P56 55 28

P57 56 27

P58 57 26

P59 58 25

P60 59 24

P61 60 23

P62 61 22

P63 62 21

P64 63 20

P65 64 19

P66 65 18

P67 66 17

P68 67 16

P69 68 15

P70 69 14

P71 70 13

P72 71 12

P73 72 11

P74 73 10

P75 74 9

P76 75 8

P77 76 7

P78 77 6

P79 78 5

P80 79 4

P81 80 3

P82 81 2

P83 82 1

P84 83 0

P85 84 1

P86 85 2

P87 86 3

P88 87 4

P89 88 5

P90 89 6

P91 90 7

P92 91 8

P93 92 9

P94 93 10

P95 94 11

P96 95 12

P97 96 13

P98 97 14

P99 98 15

P100 99 16

P101 100 17

P102 101 18

P103 102 19

P104 103 20

P105 104 21

P106 105 22

P107 106 23

P108 107 24

P109 108 25

P110 109 26

P111 110 27

P112 111 28

P113 112 29

P114 113 30

P115 114 31

P116 115 32

P117 116 33

P118 117 34

P119 118 35

P120 119 36

P121 120 37

P122 121 38

P123 122 39

P124 123 40

P125 124 41

P126 125 42

P127 126 43

P128 127 44

P129 128 45

P130 129 46

P131 130 47

P132 131 48

P133 132 49

P134 133 50

P135 134 51

P136 135 52

P137 136 53

P138 137 54

P139 138 55

P140 139 56

P141 140 57

P142 141 58

P143 142 59

P144 143 60

P145 144 61

P146 145 62

P147 146 63

P148 147 64

P149 148 65

P150 149 66

P151 150 67

P152 151 68

P153 152 69

P154 153 70

P155 154 71

P156 155 72

P157 156 73

P158 157 74

P159 158 75

P160 159 76

P161 160 77

P162 161 78

P163 162 79

P164 163 80

P165 164 81

P166 165 82

P167 166 83

P168 167 84

P169 168 85

P170 169 86

P171 170 87

P172 171 88

P173 172 89

P174 173 90

P175 174 91

P176 175 92

P177 176 93

P178 177 94

P179 178 95

P180 179 96

P181 180 97

P182 181 98

P183 182 99

P184 183 100

P185 184 101

P186 185 102

P187 186 103

P188 187 104

P189 188 105

P190 189 106

P191 190 107

P192 191 108

P193 192 109

P194 193 110

P195 194 111

P196 195 112

P197 196 113

P198 197 114

P199 198 115

P200 199 116

P201 200 117

P202 201 118

P203 202 119

P204 203 120

P205 204 121

P206 205 122

P207 206 123

P208 207 124

P209 208 125

P210 209 126

P211 210 127

P212 211 128

P213 212 129

P214 213 130

P215 214 131

P216 215 132

P217 216 133

P218 217 134

P219 218 135

P220 219 136

P221 220 137

P222 221 138

P223 222 139

P224 223 140

P225 224 141

P226 225 142

P227 226 143

P228 227 144

P229 228 145

P230 229 146

P231 230 147

P232 231 148

P233 232 149

P234 233 150

P235 234 151

P236 235 152

P237 236 153

Q4 Turnaround time :- Completion time - Arrival time

$$P_1 = 12 - 0 = 12$$

$$P_2 = 4 - 1 = 3$$

$$P_3 = 8 - 2 = 6$$

$$P_4 = 5 - 4 = 1$$

$$\text{Average Turnaround Time} = \frac{(12 + 3 + 6 + 1)}{4} = 21/4$$

$$= 5.25$$

Q6	Process	Arrival time	Execution time
	P ₁	0	8 3 2 X
	P ₂	1	7 8 3
	P ₃	3	4 X

FCFS

	0	5	12	16
	P ₁	P ₂	P ₃	P ₁
	17	03	11	103

Round Robin

Processor	0	2	4	6	8	10	12	13	15	16
	P ₁	P ₂	P ₃	P ₁	P ₂	P ₃	P ₁	P ₂	P ₂	P ₂

$$E1 = 0 + 8 = 8$$

$$P1 = 0 + 8 = 8$$

Ready	P ₁	P ₂	P ₃	P ₁	P ₂	P ₃	P ₁	P ₂	P ₂
	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)

(10) + (11) = (13) mil

(14) mil

(15) mil

(16) mil

(17) mil

(18) mil

(19) mil

(20) mil

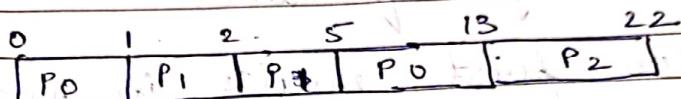
Tutorial-7



Page No.: 211

Date:

Process Name	Arrival Time	Burst Time
P ₀	0	9-8
P ₁	4	4-3
P ₂	22	8-2



$$\text{Waiting time} = \text{Turnaround time} - \text{Burst time}$$

$$\text{Turnaround time} = \text{Completion Time} - \text{Arrival time}$$

$$P_0 = 13 - 0 = 13$$

$$P_1 = 4 - 1 = 4$$

$$P_2 = 22 - 2 = 20$$

Waiting time

$$P_0 = 13 - 9 = 4$$

$$P_1 = 4 - 4 = 0$$

$$P_2 = 20 - 9 = 11$$

~~$$\text{Average of waiting time} = \frac{4+0+11}{3}$$~~

$$= 5$$

=====

Q2 Round robin is most suitable for a time-shared operating system.

- As it uses preemption based on a clock.
- An amount of time is determined that allows each process to use the processor for that length of time.

- shared operating system
 - As it uses preemption based on a clock.
 - An amount of time is determined that allows each process to use the processor for that length of time.

Date:- 18/10/19

Name:- Vinita Dayal Bhatia

Roll No:- 18MCA100

Course code:- 3CA1352

Course Name:- Operating System

Practical No:- 5(a) & (b)

Aim:- To study about finding the name of the file starting with the given alphabet from the user.

Methodology:- Shell script in UNIX Programming

```
a) if ($check -eq 1)
then
    files = `ls | grep " ^ $ch" '
    echo " Files $files"
else
    echo " something wrong"
fi
```

b) count total number of directory

```
ln = `ls -l | grep " ^ d" | wc -l'
echo " Total : $ln "
```

~~```
files = `ls -l | grep " ^ " | wc -l'
echo " Total files: $files "
```~~

Signature of Teacher:-

Date: - 18/01/19

Name: - Vinita Dayal Bhatia

Roll No: - 18MCA002

Course Code: - 3CA1352

Course Name: - Operating System  
Practical No: - 06

Aim: - To learn about head, tail, grep command with menu based.

Methodology:-

who "Enter a choice"  
read choice

- 1) head - n 4 file1.dat
- 2) tail - n 4 file2.dat
- 3) grep - n ^ t file1.dat
- 4) grep - n \$ file1.dat
- 5) nl - s " " file1.dat
- 6) grep -c -E "\^, \\$" file1.dat
- 7) Exit

Conclusion: - learnt about head, tail, cat, grep commands

Signature of Teacher :-



17

Date: - 18/10/19

Name: - Vinita Dayal Bhatia

Roll No.: - 18MCA002

Course code: - 31A13S2

Course Name: - Operating System

Practical No.: - 07

Aim: - To check only even number of arguments are allowed.

Methodology:-

num = `expr \$# . . .`

if [ \$ num -ne 0 ]  
then

echo "Only even number of arguments are allowed"

else

count = 1

while [ \$ count -lt \$# ]

do

cp \$1 \$2

shift

shift

count = `expr \$count + 2`

echo "file copied successfully"

done

fi

~~Conclusion: - learnt how to allow even number of arguments~~

signature of Teacher:-

Practical No.: - 07(b)

Name:- Ninita Bhatia (18MCA002)

Aim:- Shell script which receives two  
filenames as argument and perform  
the following operation.

echo "Enter file name"  
read file1

echo "Enter second file name"  
read file2

cmp \$ file1 \$ file2 > error

total='wc -c \$file1 \$file2 | cut -f 1 -d "

echo \$ total

if [ \$ total -eq 0 ]  
then

    echo "Both file are same"

else

    echo "Both file are not same"

fi