Subject _____ optional

Sat Sun Mon Tue Wed Thu Fri

Date :

~~Data co~~ methodology, data collection, budget & schedule, limitation, conclusion

propose evaluation metho

o

Operating System

Karnel → Heart of OS. Karnel running, OS running. CPU controlls everything.

Buffer : Temporary short memory.

Interrupt Timeline: ***

exception: user ~~দ্বারা~~ interrupt

trap : Software generated interrupt

Bootstrap program: stored in EP rom

Pooling : ~~একটুক্ষন~~ পরপর CPU নিজে গিয়ে check করে interrupt আসছে কিনা

vectored : I/O device request

interrupt Flowchart ***

Kernel: core part of OS. managing communication between software and hardware. It serves as a bridge that faciliate interaction between application and the physical resource of a computer like CPU, I/O devices

Lab

Oracle virtual box.

Pendrive

Lab

172.16.100. PC no

Subject 143

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Command

Kernel version check: uname -r

architecture : uname -m (32/64)

print working directory : pwd

list of current dir : ls

hidden files : ls -a

clear : screen clear

Directory

make directory: mkdir OS

" multiple " : mkdir A B C D. ——

change directory : cd A [inside A directory]

Back to previous folder: cd .. [Parent folder এ back যাবে]

return to home dir : cd

Home থেকে direct কোন folder এ যেতে হলে: cd /home/student

যেকোনো স্থান থেকে না চাইলে: cd ~/Desktop/A/A1 [relative path]

Desktop/A/A1 [এটাকে actual path বলে]

File make → filename

empty make file

touch f1 [by default text file]

write inside file : cat > f1 → ctrl+D [file থেকে বের হওয়া]

read file : cat f1

Lab task-01

Subject

Sat Sun Mon Tue Wed Thu

cp rename file ~/Desktop/LabP1/dir2

Date: / /

automatic file create করতে কিছু লিখতে চাইলে: cat > f2

append 2 file: cat f1 >> f2 [f2 update হবে]

2টা file কে concat করে অন্য file এ রাখা: cat f1 f2 > f3

copy file: cp f3 f4 [f3 source, f4 destination]

f3 desktop এ, f4 অন্য folder এ থাকলে path দিয়ে দিতে হবে।

rename file: mv f4 file4

file delete: rm file4, rm * [rm দিয়ে পুরো file delete করা যায়, folder যায় না]

folder delete: rm dir/ [folder name

folder del করতে হলে আগে folder খালি করতে হয়]

Lab-03

calender

Display calender: cal

Date

prev, current, next month: cal -3

current month starting from sunday: ncal -> ncal -M (Monday)

system date: date

month only: date +%m [11]

month name & month: date +%h %m [nov 11]

month name: date +%h [nov]

Minitue: date +%M [2:42] 42

Day/তারিখ: date +%d

am/pm: date +%p

কোন port এ কে connected: who

username: whoami / logname

[history, clear

Subject

Sat Sun Mon Tue Wed Thu Fri

Lec-02

Date : / /

I/O structure

system call -> any kind of message sent

*** Storage structure

- register: সবচেয়ে ছোট কিন্তু fastest [0/1]

ram = main memory [tempory data রাখে]

main memory ছাড়া computer এর কোন কিছু run হবে না। সবকিছু run হয় main memory থেকে,

fastest & small size ↑ hierarchy

cache: secondary main memory. It is also temporary memory

optical disk: cd/dvd

magnetic: caset

tertiary: extra খোলা সময় এবং বাকী সময়

Multiprocessor

throughput: ultimate result

economy of scale:

clustered system

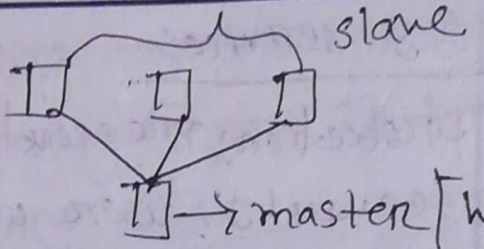
: multiple pc একটা network connected. ex: LAN

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Asymmetric clustering: ~~for~~ master, slave computer কে কাজ ভাগ করে দেয়া

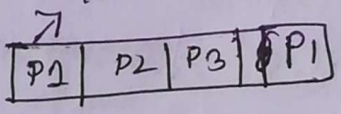


Multi programming: একটা process করে কাজ করে। FIFO [queue],

Multitasking / time sharing

P1
P2
P3

একটা certain সময় পর্যন্ত run করে। এরপর কে কাজ করে দিবে.



Broad question

***

Dual mode

Process: Program → computer যা কিছু execute করে (application). যখনই একটা program, main memory তে load হয় তখনই সেটা নাম হয়ে যায় process, main memory run হয় process হিসেবে। এজন্য তাকে activate বলা হয়।

Subject

Sat Sun Mon Tue Wed Thu Fri

Lec-03

Date : / /

OS services

Protection: Pre-caution

Security: user authentication

*** A view at os services / system call

API: Software linkup করা OS এর সাথে; যেন software এর মধ্যে interface থাকে।

Windows : Win32

Linux/Mac : POSIX

Java : JVM

লাগবে না, Standard API(X)

Linkers and loaders [*****] [exam এ আসবেই]

Linker: সবগুলো file আকারে সবগুলোকে obj file বলে। main memory তে যাওয়ার আগে linker সবগুলো file কে combine করে একটা execute file বানায়। then loader main memory তে load করে।

Microkernel: Kernel mode থেকে shift করে user mode এ নেয়া। example: darwin

Subject: form: Process creation ও system call
Lec-04
Date: / /

OBE

Process state

new
running
waiting: যতক্ষণ না i/o user থেকে পাচ্ছে ততক্ষণ wait করবে.
ready: সবকিছু নিয়ে ready. CPU তার কাছে running ও চলে যাবে.

waiting থেকে কখনো direct running ও যাবে না ready হয়ে যেতে হবে.

*** PCB: process related every information store করে.

CPU switch from process to Process / context switch

Short time scheduler: যে ready থেকে running ও নেয়.

CPU scheduler / short term

. ready → running [Preemptive]

long term / job

control degree of multiprogramming.

which process should be brought into the ready queue.

Subject Command
Sat Sun Mon Tue Wed Thu Fri
Date: / /

Lab-3

sort f2 | nl | cat > f4

Lab-4

arithmatic operation: bc   5+5=10

word count: wc f1   2 → line no 5 → word count 20 → char count f1 → file name

only line no: wc -l f1   wc -c f1, wc -w f1

ascending order list: nl f2 →
1. banana
2. apple
3. guava

increment by 5 : nl -i5 f2

multiple command কে জোড়া লাগানো এবং অন্য file এ
save: nl f2 | cat > f3

A-Z : sort f2   Z-A: sort -r f2

remove duplicate: sort -u f2

Lab-4

file এর 1st 10 টা name: head f2
head -5 f2 (1st 5 টা line)

last 10 টা name: tail f2   tail -5 f2

delemator

cut -d " , " -f 1,3 f5 → field
→ 1 & 3rd column
1-2 [1 থেকে 2]

কোন line এর char cut: cut -c 1-3 f5

file কে copy করা column wise / table marge
: paste f5 f2

Subject

lab-5

Sat Sun Mon Tue Wed Thu Fri

Date : / /

search: grep "test" f1 → line সহ highlight

by default → case sensitive ← insensitive // this is a test

case insensitive: grep -i "test" f1

count pattern: grep -c "test" f1

grep -i -c "test" f1 // insensitive

only pattern: grep -o "test" f1 // test

reverse search: যে line এ pattern (test) নাই, সে line দেখাবে।

grep -v "test" f1

4:00

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Lec-6

CPU scheduler

preemptive, Non preemptive

scheduling criteria

throughput: per second G

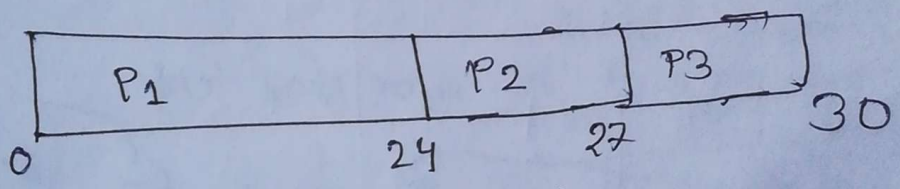
waiting: ready state. CPU এর জন্য wait করে।

Turnaround time: process create হওয়া থেকে terminate হওয়া পর্যন্ত system এ যতক্ষণ থাকে।

*** First - come, First served

Burst time: Process run/execution হতে যে সময় লাগে

time শুরু হবে 0 থেকে।



waiting time: $P_1 = 0$, $P_2 = 24$ $P_3 = 27$

averaged waiting time = $(0+24+27)/3 = 17$

Turn around time = waiting time + burst time

$= (0+24) + (24+3) + (27+3) = 81$

average turnaround time = $81/3 = 27$

Subject _____

Sat Sun Mon Tue Wed Thu Fri
○ ○ ○ ○ ○ ○ ○

Date : / /

convoy effect: short process behind long process

shortest job first → যার burst time কম, তা আগে যাবে।

shortest remaining time first

quiz আসবে

$P_1 = 8\ 7$

$P_2 = 4\ 2$

waiting time

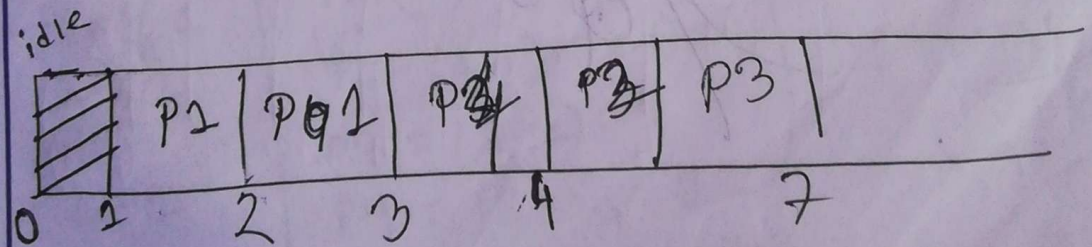
$P_1 = 0 + (10-1) = 9$

$P_2 =$ entry − arrival $= 1-1 = 0$

$P_3 = 17 - 2 = 15$

$P_4 = 5 - 3 = 2$

Process	arrival	Burst
P1	1ms	2ms 1 0
P2	4ms	1ms 0
P3	3ms	2ms
P4	2ms	1ms 0



P1 = (1-1) = 0 P2 = 4-4 = 0 P3 = 5-3 = 2 P4 = 3-2 = 1

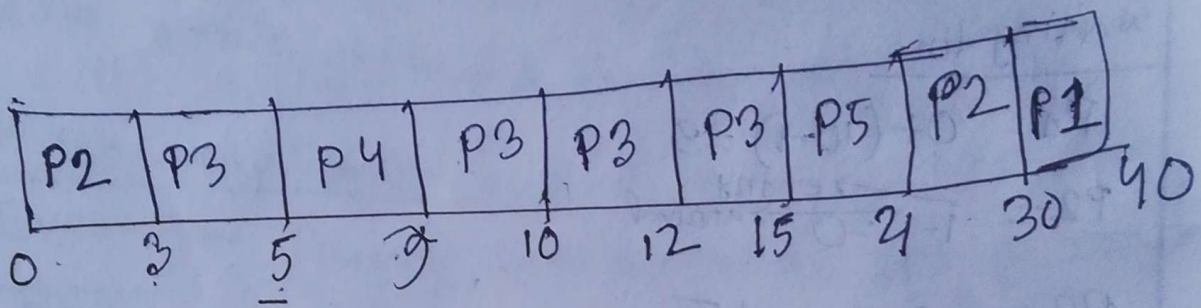
Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Process | Arrival | Burst time

Process	Arrival	Burst time
P2	0 ms	12 ms
P3	3 ms	8 ms
P4	5 ms	4 ms
P1	10 ms	10 ms
P5	12 ms	6 ms



P1 = 30 - 10 = 20

P2 = 0 + (24 - 3) = 18

P3 = (3 - 3) + (9 - 5) = 4

P4 = 5 - 5 = 0

P5 = 15 - 12 = 3

(20 + 18 + 4 + 3) / 5

Most important

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

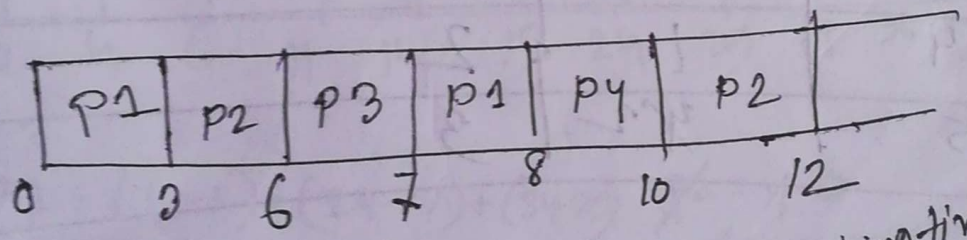
Round Robin → multitasking : Burst time compare হবে না।

time quantum তোমে ready queue তে last এ পাঠিয়ে দিবে।

P1 5 3 → time quantum
P2
P3
P1

# Process	Arrival time	Burst time
P1	0	~~4~~ 1
P2	1	5
P3	2	1
P4	5	2

Time quantum = 3 ms

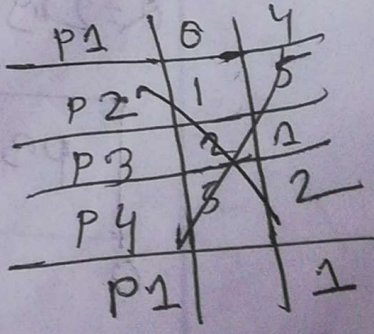


Ready queue

P1 = ~~4~~ 1
P2 = ~~5~~ 2
P3
P1 = 1
P4 =
P2 = 2

P1 = (0-0) + (7-3) = 4
P2 = (3-1) + (10-6) = 6
P3 = (6-2) = 4
P4 = (8-5) = 3
avg. waitime = $\frac{4+6+4+3}{4}$

waiting time



Turn around = wt + Bt

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Priority Scheduling

যার number কম তার Priority বেশি।

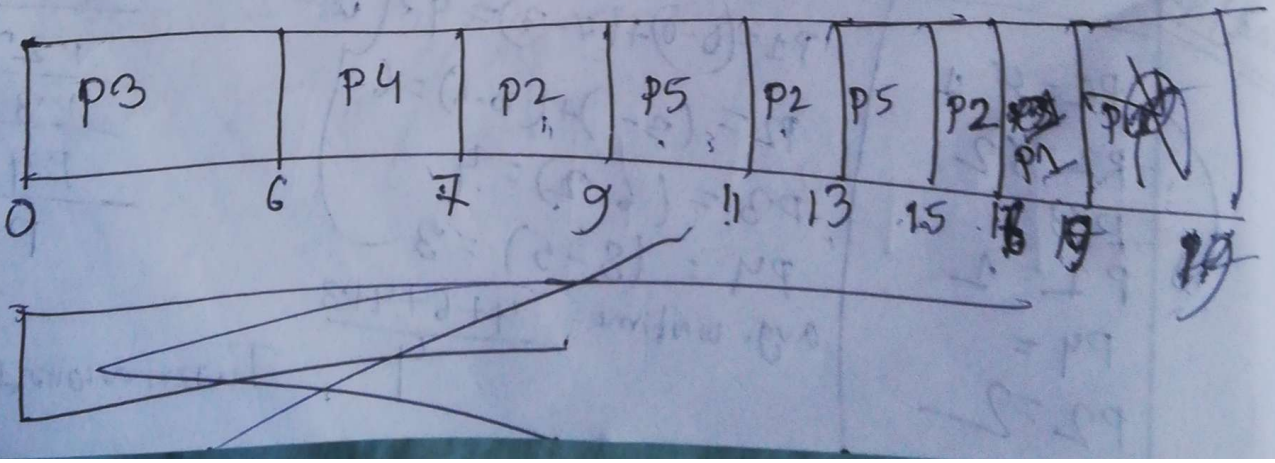
Starvation : low priority process may never execute

Aging : যাদের lowest priority, OS তাদের priority বাড়ায় দিবে।

যখন multiple process এর priority same হবে, তখন তাদের মধ্যে round robin apply করতে হবে।

Process	Burst time	Priority
P1	3	5
P2	6 ~~2~~ 2	3
P3	6	1
P4	1	2
P5	4 2	3

Q=2ms



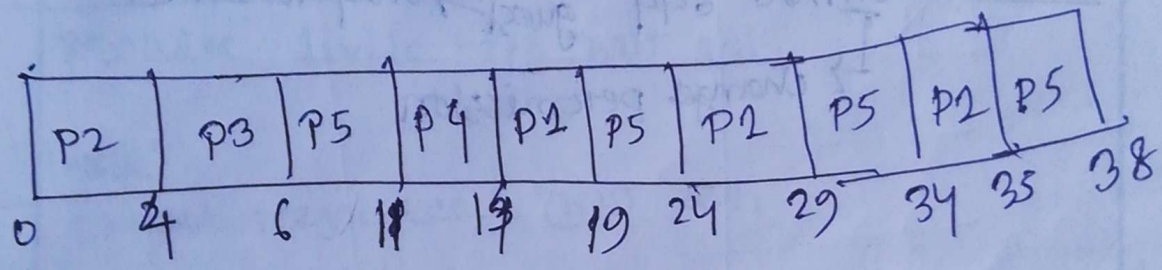
Subject

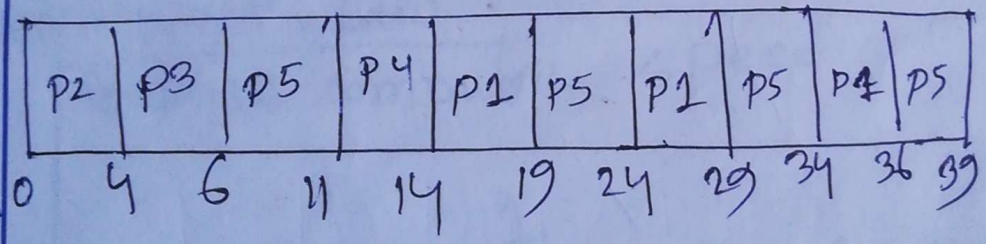
Sat Sun Mon Tue Wed Thu Fri

Date: / /

quantum time = 5

Process	Arrival time	Burst time
P1	5	12
P2	0	4
P3	0	2
P4	3	3
P5	2	18





P1 = (14-5) + (24-19) + (34-29)
= 19

P2 = 0

P3 = 4

P4 = (11-3) = 8

P5 = (6-2) + (19-11) + (29-24) + (36-34) = 19

P2 = 4 0
P3 = 2 0
P5 = 18 13
P4 = 3 0
P1 = 12 7
P5 = 13 8
P1 = 7 2
P5 = 8 3
P1 = 2

P2 = 4 0
P3 = 2 0
P5 = 18 13
P4 = 3 0
P1 = 12 7
P5 = 13 8
P1 = 7 2
P5 = 8 3
P1 = 2 0
P5 = 3

P5 = 3

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

lab-5 : file permission

Ownership of Linux files

user: file/folder এর owner

group: একাধিক user নিয়ে group create হয়।

r(4) w(2) x(1) → execute

421 → যারা program run করতে পারবে।

user → group → others এর value

command: chmod 644 guest → file/folder

↑ user, group, others

↳ change permission

Subject

Theory- Le

Thread : s

Thread / Light

Benefits

Parallelis

quad core

Data parall

Resource

Task

shares

Amdahl

measure

spe

Multi th

user

Kernel

Subject

Sat Sun Mon Tue Wed Thu Fri

Theory- lec-5 flow Date : / /

Thread is single sequence of ~~instruction~~ activity.

Thread / lightweight process

Benfits

*** Parallelism : can perform more than one task simultaneously.

quad core system ৫ ৪টি thread চালাতে পারবে

Data parallelism

Resource divide করে নেয়া হয়।

Task :

shares resource নেয়া হয়।

*** Amdahl's Law

measure computer speed

$$\text{speedup} \leq \frac{1}{S + \frac{(1-S)}{N}} \to P$$

$(1-P)$

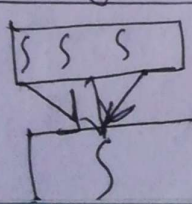
S = system serial portion এর amount

(1-S) = Parallel portion

N = no of core

*** Multi threading model

user



kernel

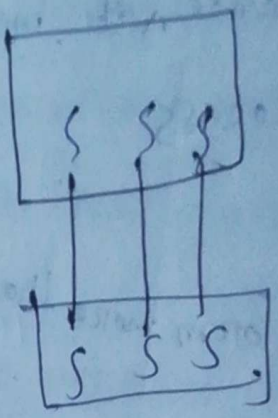
(many to one)

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

One to one



Thread pool

আগে থেকে server G thrade create 2000।

Fork join

Mid Question Pattern

Part-A

mcq from shell cmd [10x2=20]

Part-B

OBE: 3 Question [3x5=15] → theory

Part-C

shell cmd writing [3x5=15]

Part-D

short Q [4x5=20]

Part-E

3 out of 4 [3x10=30]

Lab-01 Subject

Sat Sun Mon Tue Wed Thu Fri

Date: / /

Final Term

vi editor → for shell programming. [visual editor]

2 mode → 1) command
2) insert mode

by default command mode এ থাকে। → for type

file save করতে হয়। command দিয়ে vi editor case sensitive.

command → insert [press i]
← [esc]

file create

vi f1 [filename]

save: :wq [save & quit] :w → save :q → quit

a[i]b → cursor i এ আছে। এখন i press করলে cursor এর আগে থেকে লিখা শুরু হবে। [aiabcib]

[capital I দিলে line এর শুরুতে নিয়ে যাবে এবং একই সাথে insert mode এ নিয়ে যাবে।]

small 'a' দিলে cursor এর পরে থেকে লেখা শুরু হবে।

ai[i]b → aiiaaib

capital 'A' দিলে line এর শেষে নিয়ে যাবে এবং insert mode এ নিয়ে যাবে।

small 'o' → new line create হবে এবং insert mode এ নিয়ে যাবে

capital 'O' → যে line এ আছি, তার উপরে

Subject ______ Sat Sun Mon Tue Wed Thu Fri Date: / /

newline create হবে এবং insert mode এ নিয়ে যাবে.

virtual linux এ up/down key কাজ করে command mode কিন্তু actual linux এ করে না.

J → ↓   K → ↑   h → ← (left)   l → (right)

backspace/sing char delete: x [press x] → cursor যেখানে আছে।

X → cursor যেখানে আছে, তার আগের char delete হবে

dd → cursor যে line এ আছে, puro line delete হয়ে যাবে

D → cursor যে char এর উপরে আছে, ঐ char থেকে right এর সব char delete হয়ে যাবে.

[nadim] → [na]

line copy: yy → Paste: P → নিচে paste হবে, P দিলে উপরে paste হবে।

Shift+A → line এর শেষে যাবে.

Subject: Lec-07 Theory

Date: / /

29/12/24
quiz - 1
Lec: 7,8,9

Multi-processor scheduling

common ready queue: সবগুলো একটি queue তে থাকে।

CMT [short qustion]

thread, processor এর মত কাজ করে।

On a quad-core system with 2 hardware thread per core, OS sees 8 logical processor

Load balancing: attempts to keep workload evenly distributed.

Push migration: periodically processor check করে, কি পরিমাণ work load কি আছে? একজনের বেশি, আরেকজন কম / idle হলে, যার বেশি আছে তার থেকে বাকিগুলোকে workload দেয়,

Pull: idle processor, যার workload বেশি তার থেকে নেয়,

Soft real-time system: Priority highest, but no gurantee when task will be scheduled. (critical)

Hard real-time system: task must be serviced by its deadline.

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

delay

*** Interrupt latency: interrupt come, service provided এর মধ্যবর্তী সময়কে interrupt latency বলে।

** Dispatch latency: $P_0 \rightarrow P_1$ [সাময়িকভাবে time delay কে Dispatch latency বলে] → context switch

Algorithm evaluation

যার average waiting time কম, সেটা best.

FCFS: যেটার task হলে কম time নিবে best হবে।

Little's formula

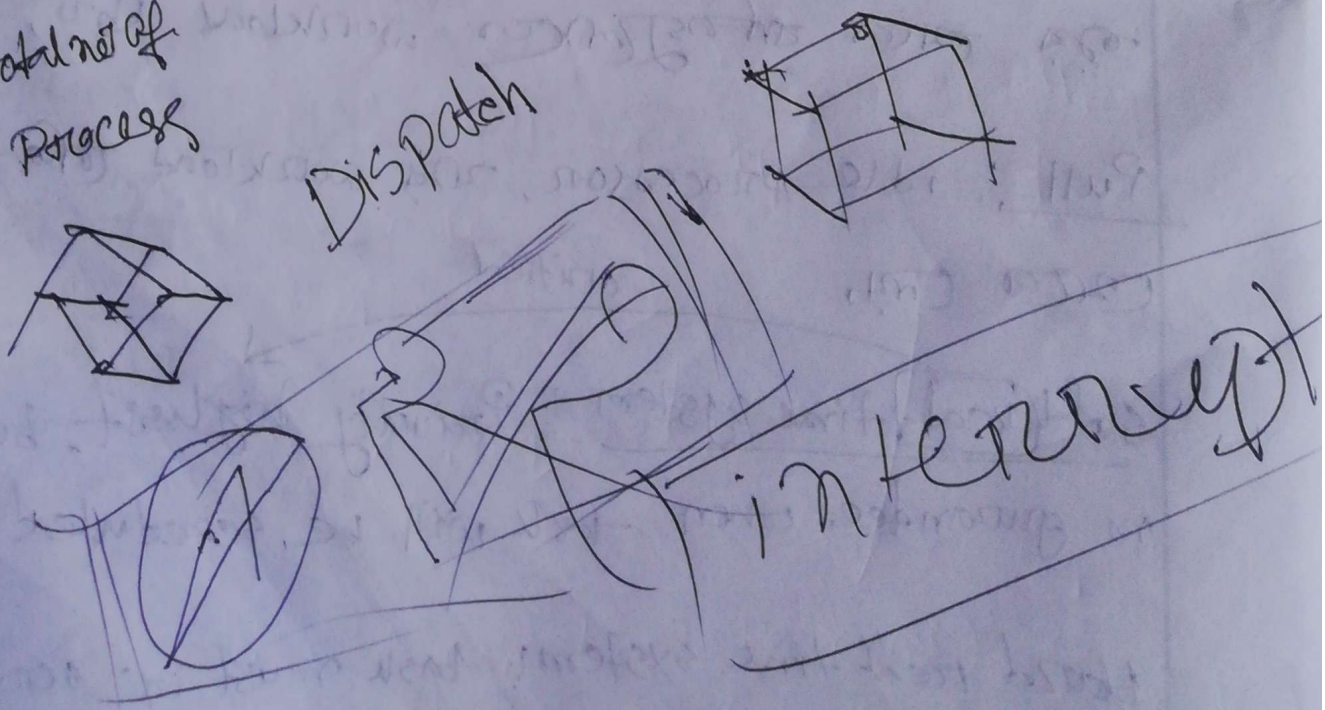
$n = \lambda \times w$

n → Process arrive per second

w → avg. waiting time

$14 = 7 \times w \quad \therefore w = 14/7 = 2$

total no of process



Subject (Hardest chapter)

Lec: 8

Sat Sun Mon Tue Wed Thu Fri

Date: / /

Synchronization Tools

Background:

consumer → ভোক্তা

producer → product produce করবে.

produce product → keep in buffer

count++

count-- [consumer take product]

Pseudocode of producer & consumer

Race condition (understand problem).

[shared data]

২টা core x নিয়ে কাজ করছে।

thread 0: x++ thread 1: x-- [at the same time]

Ans: we don't know

code এর যে part এ race condition occur হয় → critical section

Soln of critical section

1) **Mutual exclusion:** একটা process যদি critical section এ থাকে, অন্য process ঢুকতে পারবে না, (OS যেকোন একজনকে permission দিবে)

2) **Progress:** critical section free, right now, immediately অন্য Process কে critical section এ ঢুকার permission দিতে হবে।

Subject ____________ Sat Sun Mon Tue Wed Thu Fri Date: / /

3. Bounded waiting: check critical section → খালি কিনা? খালি see waiting list. হলেও আগে থেকে কেউ ঢুকতে পারবে না

lab-02

#! /bin/bash → interpreter location [প্রত্যেক code এর শুরুতে থাকবে]

Print: echo "Hello World" → আগে commands mode এ যেতে হবে।

save & exit: Press esc → :wq

run ./filename [chmod 777 f1] → filename

Shell variables

- there is no data type for variable নিজে থেকে identify করে নিবে।
- variable যখন call/use করবো name এর আগে $ use করবো। $a.
- numeric দিয়ে variable name শুরু হবে না, name এর মাঝে space দেয়া যাবে না।

ex: go to insert mode

```
name=Nadim [no space]
echo "The name is " $name
press esc → :wq
chmod 777 f1
./f1
```

user input: read name → (variable name)

read v1 v2 (run এর সময় space দিয়ে দিয়ে input দিতে হবে)

Subject _____

Sat Sun Mon Tue Wed Thu Fri
○ ○ ○ ○ ○ ○ ○

Date : / /

newline : /n comment : #

read -p "Enter the name" name → variable

[read -s] → silent → newline

read -sp → same line

Operator (+ - * /)

operator এর আগে পরে space অবশ্য mandatory

a=20 b=10

sum = $((expr $a+$b))

sum= `expr $a + $b`

sum=$(expr $a+$b) → no space

for multiplication: **

condition: [ ]

Floating point → কোনো space হবে না। bc এর আগের পরে space হবে।

1) echo "$num1+$num2" | bc

2) num3 =$(bc <<< "$nam1 + $num2")

```
read a b
sum=$(bc <<< "$a+$b")
echo $sum
```

floating point এর multiplication এ (*) চিহ্ন ব্যবহার হবে না।

theory Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

lec-8

Peterson's solution

why it is not work good

=> 2 shared data //th1, th2 shared data access করে?

thread 2

swap flag = true } x এর value তে print হবে,
x = 100 } তা আমরা জানি না। এইজন্য modern architecture এ
P.S ভালো কাজ করে না।

lec-9

Hardware instructions

test and set instruction : Parameter এর original value return করে।

Soln:
lock = false

P1, P2 → Process

~~P2 loop এর ভিতরে আটকে থাকবে, যতক্ষণ P1~~
~~এর~~

P1 → critial section (test and set (false)
return false
loop condition false,
P1 critical section এ ঢুকে গেছে,
P2 যখন ঢুকতে যাবে তখন lock true. loop
এর ভিতরে আটকে যাবে যতক্ষণ না P1 critial

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

section G ঢুকে আর কেউ ঢুকতে,

while (test and set (lock))
/* do nothing */

/* Critical section */

Compare and Swap

parameter হিসেবে original value return হয়,

lock = 0 (initially)

P1 loop থেকে বের হবে।

P2 আসছে, lock এর value 1
expected 0, new value 1
temp = 1 (condition false)
while condition true
P2 loop এ বারবার ঘুরতে থাকবে, যতক্ষণ না P1 এর কাজ শেষ হচ্ছে।

Atomic variables: uninterrupted
// it's like compare & swap.

short question → Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Mutex lock & → Binary semaphore

acquire () {
while (! available) ← lock = true
// busy

available = false // loop থেকে বের হলে
}

// enter critical section

release () {
available = true
}

Semaphore → broad question

wait () , signal ()

counting semaphore

Ref

25376

10)25376(

6

0+6 = 6

Subject _____

Sat Sun Mon Tue Wed Thu Fri
○ ○ ○ ○ ○ ○ ○

Date : / /

lab-3

```
#!/bin/bash
read num
if [ $num -ge 0 ]
then
echo "$num is positive"
else
echo "$num is negative."
fi
```

→ a number is greater or equal to 0.

ending of if else block

else if

if, elif

switch case

break = ;;

default = *

while loop → reverse an int number

```
read n
rev=0
while [ $n -gt 0 ]
do
rem=$(expr $n % 10)
rev=$(expr $rev \* 10)
rev=$(expr $rev + $rem)
n=$(expr $n / 10)
done
echo $rev
```

lab-4

Loop

```
read num
for (( i=0; i<=num; i++ ))
do
echo $i
done
```

```
for number in {1..10}
do
echo $number
done
```

obe

1 2 3 4 5 ...10

obe

```
# for number in 1 2 3 4 5
do
echo $number
done  // ans: 1 2 3 4 5
```

1
2
3
4
5

Subject (Theory)

Lec: 10

Sat Sun Mon Tue Wed Thu Fri

Date: / /

min. 40 marks will come from this chapter

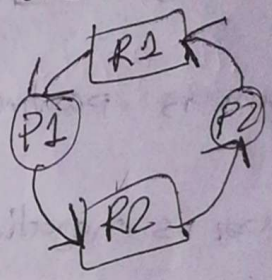
Deadlocks (**********)

instances: resource এর যতগুলো copy আছে

deadlock:

mutex=resource

th1, th2



th1 has first resource, but it need second resource

th2 " second " , " " " first "

th1 has no access of second resource

th2 " " " " first "

task can't be completed. this is deadlock

Characterization

problem of deadlock

1) mutual exclusion: only one process at a time can use a resource.

2) Hold and wait: একজন কে নিয়ে রাখছি, আরেকজন wait করতেছে।

3) No preemption: resource যে process কে দিয়ে রাখছি, শুধু সেই release করতে পারবে, অন্য কেউ পারবে না।

4) Circular wait: there are many process ($P_0 \dots P_n$)

P_0 একটা resource এর জন্য wait করতেছে যেটা P_1 রাখছে

Subject: Resource never destroy,

Sat Sun Mon Tue Wed Thu Fri

Date : / /

P1. same for others.

P_i / T_i Process / Thread

Resource Allocation Graph (1টা question আসবেই)

request edge : $P_i \rightarrow R_j$ [Process request করেছে]

assignment : $R_j \rightarrow P_i$ [resource already process এর কাছে assign হয়ে গেছে]

• → instance

: → R2 [at a time R2 এর 2টা নিয়ে পারবে]

Basic facts

single instance এ cycle হলেই deadlock. but multi instance এ না হতেও পারে

no cycle → no deadlock

graph contain cycle

1) per resource has only one instance → deadlock

2) several instance per resource → Possibility of deadlock

Deadlock prevention

Mutual exclusion: make shareable and readonly

Hold and wait: যখন কোনো process একটা resource কে request করবে, তখন কোনো resource hold করে থাকা যাবে না, request এর আগেই ছেড়ে দিতে হবে।

No Preemption: request করবে, request এর পর immediate resource না পেলে release করার পূর্বে নতুন resource পাওয়ার পর আগেরগুলো ছেড়ে দিবে।

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Deadlock avoidence

Safe state : no probability of deadlock | unsafe : probability of deadlock

P1, P2, P3 ... Pn

P3 এমন সব resource কে request করতে পারতে যেগুলো available আছে. P1, P2 কেও পারতে কিন্তু P4, P5.. কে পারতে না,

Resource allocation graph

claim edge : $P_i \rightarrow R_j$ (----->) : এখনো request করে নাই but may request in future. যদি request করে তখন Process edge এ convert হয়ে যাবে.

Data structures for Banker algorithm

available : resource ^ system এ available আছে কিনা ?

max : প্রতিটা resource যে max কতগুলো copy request করতে পারে

Allocation : Process, resource যত গুলো copy নিয়ে রাখছে,

need : (max - allocation)

Process, resource

Subject: echo = newline

Sat Sun Mon Tue Wed Thu Fri

Date: / /

1
3
5
7
9

lab-4

for number in {1..10..2}
do
echo $number
done

(O/P trace)

for output in $(pwd)
do
echo $output
done (Learns: home/student/desktop)

Lab-5 : array

os=(ubuntu windows kali)
echo "${os[0]}"
echo "${#os[@]}" → array size

input:
read -a array
echo "${array[1]}"

File parsing (OBE)

while read line
do
echo $line
done < f1 → line by line parse করবে।

Theory

Bankers algorithm

sol":
need = max-allocation
(A B C)

need:
P0 = 7 4 3
P1 = 1 2 2
P2 = 6 0 0
P3 = 0 1 1
P4 = 4 3 1

Safe/unsafe

A = available
AL = Allocation

P0 = const
P1 = 1 2 2
A = 2 1 0

P1 এখানে কোন যা return করে দিবে। (need: 1 2 2, available: 2 1 0, allocation)

new available
A = 5 3 2

IFS = internal field separator (cut এর মত)

while IFS '@' read id extra
do
echo $id
done < f2

existing chk of file

filename = f2
if [-f $filename]
then
echo "file exist"
else
echo "file not exist"
fi

Subject

Sat Sun Mon Tue Wed Thu Fri

Total available + total allocation = total instances Date : / /

#

Process	Allocation				max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	4	0	1	3	7	2	3	4	2	1	3	3
P1	0	1	1	2	3	7	5	6				
P2	0	0	3	4	2	1	5	6				
P3	2	3	5	2	3	4	6	6				

Is the system in a safe state? Justify your answer by applying the Bankers algorithm

need = max - allocation

P_0 = 3 2 2 1	P1 = 3 6 4 4
P2 = 2 1 2 2	P3 = 1 1 1 4

safe/unsafe: A = available AL = allocation

A = 2 1 3 3

P2 = 2 1 2 2

A = 0 0 1 1

No need can be satisfied. return all resources consumed by P2.

A = 0 0 1 1 + 2 1 2 2 = 2 1 3 3

A = A + AL = 2 1 3 3 + 0 0 3 4 = 2 1 6 7

P3 = 1 1 1 4 | No need can be satisfied return all resource consumed by P3.

A = 1 0 5 3

Subject ____________ Sat Sun Mon Tue Wed Thu Fri Date: / /

A = A+AL
= 2 1 6 7 + 2 3 5 2 = 4 4 11 9

P_0 = 4 0 1 3
A = 1 2 9 8

A = A+AL = 5 2 10 11 + 4 0 1 3
= 9 2 11 14

Request check

1. To check request valid কিনা? (need এর চেয়ে ছোট হলে valid)
2. available কিনা?

Memory management

base: starting

Logical vs Physical Address ↗ exact location

Physical / mac address: actual address. unchangeable.

logical address: CPU generate it. runtime address. Program যতক্ষণ চালু থাকবে, ততক্ষণ থাকবে,

Swapping

virtual ram / Backing store: screen এ যতগুলো application open থাকে use না করলে vram এ থাকে।
Ex: open → teams, browser, vs code, ppt. ppt use করা হচ্ছে, বাকিগুলো না। ppt থাকবে ram এ, বাকিগুলো vram এ

Subject _____

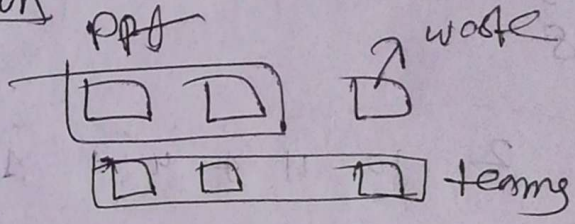
Sat Sun Mon Tue Wed Thu Fri
○ ○ ○ ○ ○ ○ ○

Date : / /

Swaping with Paging

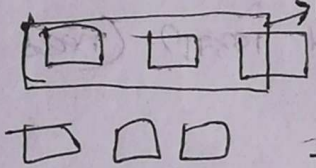
Backing store তে memory block গুলোকে pages বলে।

Fragmentation

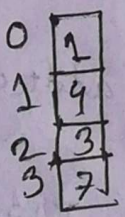
External: 

Internal:

compaction → sort of external fragmentation.

Internal: 

Paging Model of Logical and Physical memory



logical এর 0 থেকে শুরু, Physical এর 1 এ।

m → total logical এ কয়টি, $m=4 \rightarrow 2^4 = 16$

n → প্রতি block এ কয়টি, $n=2 \rightarrow 2^2 = 4$

n → কয়টি block হবে, $n=2 \rightarrow 2^2 = 4$

Subject

Lab-6

Sat Sun Mon Tue Wed Thu Fri

Date : / /

Function

no return type

```
#! /bin/bash
fact () {
f=1
for ((i=1; i<=$1; i++))
do
f=$ (expr $f \* $i)
done
echo $f
}
result=$(fact $(fact 3))
echo "factorial = " $result
```

first parameter

Print হবে না, return হবে।

Protection

Object

hardware: Printer, - - -

software: file, program, semaphore

Domain = multiple user নিয়ে domain তৈরি হয়।

= set of access-rights

*Access matrix

Security

Intruders: outsider (system এ authorize user বাদে বাকি সব)

Breach: ভাঙা (rules ভাঙা)

Subject

Sat Sun Mon Tue Wed Thu Fri

Date : / /

confidentiality: data read, no change (without permission)

availability: destroy avl. data

Theft of service:

Man in the middle attack: Hello to buy করে fish.

encryption:

malware: software generated.

Trojan horse

i) spyware: website add

ii) Ransonware: Hide data

Virus dropper: insert virus onto the system

DDoS: attacker ↓ server to target করে. continous করে।
authorize user req করে কাজ করতে server down.

In DDoS: group of attackers. multiple site থেকে req করে / bot create করে.

Q.A

A-: Shell program writing (2 out of 2) 5×2=10 ~~2.5×2=5~~

B- OBE (Shell command writing using system variables)

↳ for loop → 5×3=15 (file parsing, IFS, existor not)

C- short question (3 out of 3) → 15

Subject ↑ math, graph sure ... Sat Sun Mon Tue Wed Thu Fri

D - Broad Q (4 out of 3) 20×3 = 60 Date : / /

memory management, synchronization → Quize 1

	Allocation				Req				Avl			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	1	1	0	0	0	1	0	0	0	1	0	0
P2	1	1	0	0	0	0	1	0	1	2	0	0
P3	0	0	0	0	0	1	1	1				
P4	0	0	1	1	0	0	1	0				
P5	0	0	1	0	1	0	0	0				
	2	2	2	1								

4 out of 3

CN, ACN
ON

1 2 0 0