

Sub:-

Sat Sun Mon Tue Wed Thu Fri

Date: 16 / 09 / 2023

Operating System

* বানানো আনেক বর্ষ। For this android এর competition নাই,

Main things { operating computers core components hardware / software } → Architecture (Hardware)

Input/output operation → computer কীভাবে connect
peripheral → operate করাবে, (Hardware)

Operating system → Software / Higher Level.

computer hardware এর মাধ্যমে software এর interface

* efficient manner এর hardware রে operate

* OS এর নির্ভুল বিট্ট program আছে। যেখানে আজনে user এর ব্যাপে visible না।

* OS এর main user root user. OS এর core component services run করবাই।

Core component → Kernel mode.

must perform করতেই
হবে প্রয়োজন operation
প্রয়োগ।

Sat Sun Mon Tue Wed Thu Fri

Date: / /

* যাব program user interface এর মাধ্যমে চের

মাথি:

Computer, এর সাথে communication ২ টা প্রকার হ

- UI
1. shell
2. GUI

* User interface-এর উপরের layer of Unix

user layer (গোলাম)

kernel layer (OS)

4 Components of Computer System

1. Hardware → CPU, I/O devices

2. Operating System

multiple user মাঝে hardware components

এর মাধ্যে control and coordinate করে।

3. Application programs

4. User → people, machine, other computers.

Operating System Concepts → Silberschatz (9th)

Modern Operating System → Tanenbaum.

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: 18/09/2023

OS এর role ২টি: Performance optimization এবং

(i) Resource allocator \rightarrow resource allocate

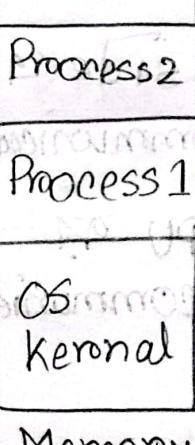
* CPU Time

* memory time (কোনো processor কে যে স্মেল্ল করে বলে)

Loader program দ্বাৰা OS এ load কৰাৎ। Windows এ
ফোন বলো Task. Linux এ বলো process. এই একটা address
space আছে। OS দ্বারা program কে time fit কৰি,
space পাওয়া।

(ii) Control Program \rightarrow To avoid illegal memory
access

\rightarrow prevent errors



int *ptr

ptr = 0x01123

হতে পাবে অন্য

int b = *ptr

বেঁচে থাকা process

এর address

\Rightarrow OS এর বিশিষ্ট
protection নেই

bootstrap program: OS boot হয় এটাৰ মাধ্যমে
EP ROM / EEPROM (Electrical Erasable peripheral)

ROM \rightarrow Mostly Read Maximum read; write least

Sat	Sun	Mon	Tue	Wed	Thu	Fri
0	0	0	0	0	0	0

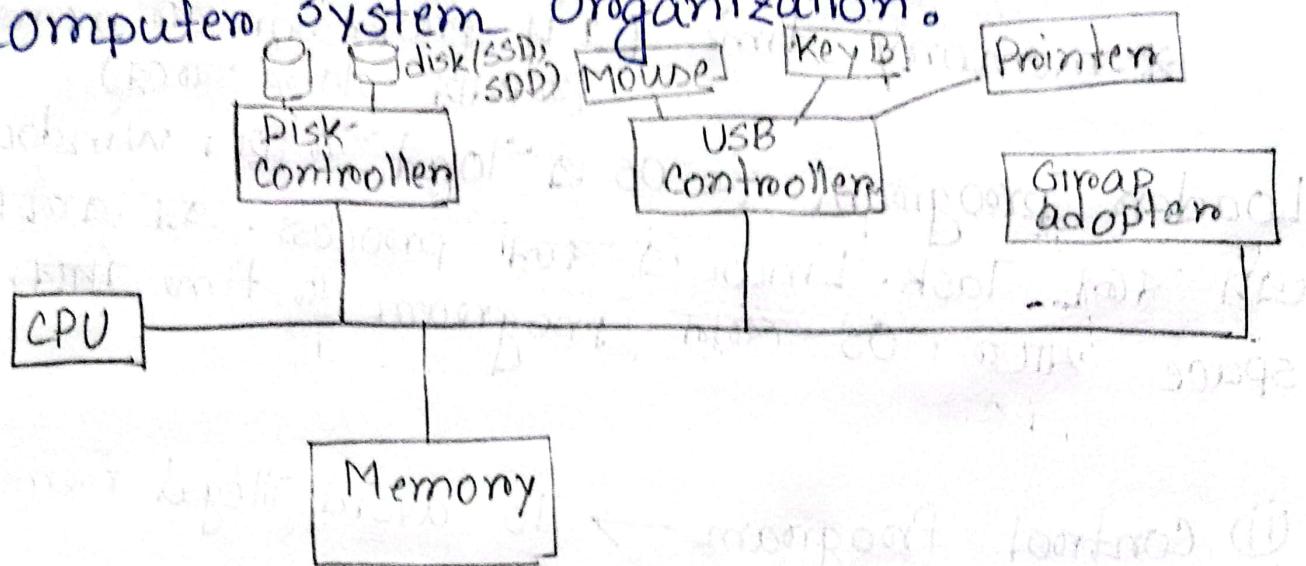
Sub:

Date: / /

OS Lower memory to load

OS hard disk এ মাত্রে। Hard disk \rightarrow permanent

Computer System Organization



I/O device \neq CPU \rightarrow concurrent execution

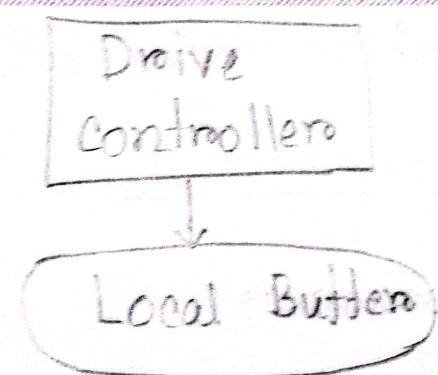
Device controller \rightarrow I/O side এ device side এ
CPU এর মাধ্যমে I/O communication.

Drivers: Device controller এর মাধ্যমে CPU এর
connection এর জন্য CPU এর communication
এর জন্য CPU side এর device

OS এর উপর একটা software

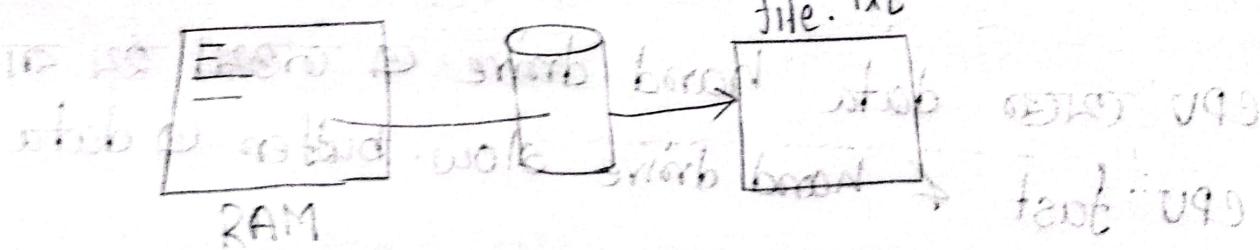
Sat Sun Mon Tue Wed Thu Fri

Date: / /



Buffer : Buffer হলো Temporarily

কোনো কিছি রাখা যা
পরবর্তীতে change হবে,
আবে অন্য আপ্রসার
Save ব্যবহাৰ।



RAM এর operation → read/write সক্ষম
I/O device "bus" → input/output

RAM → Volatile, PC off কৰলে erase হয়ে যাবে,

Text editor এ কিছি লিখলে তা RAM এ save হব।
পরে তা hard drive এ load কৰলে ব্যবহৃত হবে।
ব্যাবহৃত RAM volatile . মেইন Hand drive থেকে
permission নিয়ে edit এর সময় RAM এ save
কৰা হয়েছিলো। আবার বাজি ক্ষেত্ৰে hand drive
এই ক্ষেত্ৰে ফিলাই তাহলে RAM এখন buffer.

Sat Sun Mon Tue Wed Thu Fri
 Date: / /

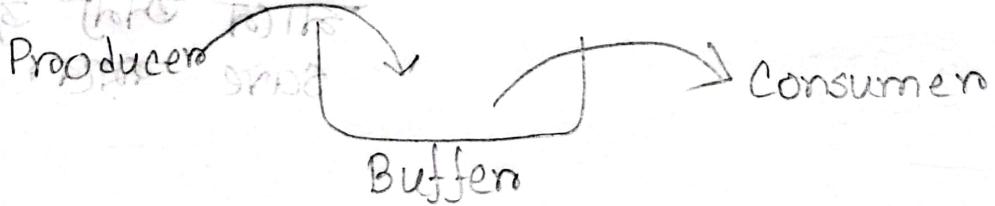
Sub:-

Producers - Consumers Problem:

TF ১৫৪ পৃষ্ঠা (৮) টপ

১০০ প্রসেসর প্রক্ষেত্র

বিনামূলে তৈরি করা



CPU যথেক্ষণে data hard drive এ জওয়াহ হয় কারণ CPU fast & hard drive slow. buffers এ data store থাকে।

Device controllers এ local buffers থাকে। local buffers মাত্রে CPU যথেক্ষণে hard drive এ data পাচাই, অনেক set slow. তাই hard drive এর processing স্টেট instant পাচাই local buffers মাত্রে processing স্টেট overflow না হওয়া পর্যন্ত local buffer process করে।

Memory → Local

Device

Controller

CPU কে I/O operation এর জন্য block বলা হলে computer hang হয়ে যেতে পারে।

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Data আদান-প্রদান: Local Device Controller Buffer
 Memory → Local Device Controller Buffer
 (CPU এর চেয়ে অনেক slow)

CPU ২টা বাতি বরতে পারে:
 (i) CPU slow perform → Local...
 Buffer
 এবং মাঝে তাল নিলিখে → এটি
 desirable না

Ex: single core device থাকলে → ১টা CPU
 CPU যদি stop হয়ে থাকে I/O operation এর জন্য
 block করা হয় → program in response হবে,
 hang হয়ে যাবে।

CPU কে তখন I/O এর জন্য idle করা হবে যখন
 I/O operation থেকে ছোট (কম),
 Solution → interrupt যা I/O operation শেষ
 CPU কে notify করবে।

✓ Sat Sun Mon Tue Wed Thu Fri

Date : 20/09/2023

Sub: _____

Storage & Definitions and Notation Review

* Computer 8 bit করে address করে,

* 32 bit OS জানে প্রত্যক্ষ address এ 32 bit
use করে,

* 4 GB র বাহরে RAM access করতে পারেন।
32 bit OS তাই 1.6GB নিলেও 4GB বাই-
বাকি ছিলো use করতে পারেন।

* 64 bit Huge number OS এর জন্য!

* Bus 64 bits হলে OS 32 bit হলে
support করবেন।

Main Memory

Random Access
একই সংখ্যা-
time লাগাবে অব-
access এর জন্য।

Secondary Memory
Volatile
hand disk
SSD
chache

* Speed এর হাল cost এর হাল non volatile.
আব cost এর হাল volatile.

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

speed বান্ধা cost বান্ধা non-volatile → Hard Drive

* hardware cache এতো fast

* registers এবং memory খুব বান্ধা।

cache memory র 3rd phase:

} L1

{ L2

L3

Registers

cache

Main memory

A ↓
solid-state disk

hard disk

↓
Optical disk

of quick access
Magnetic tape

sequential

Storage
Data archive
use করা হয়।

* Multiple Redundancy রাখে cloud computing এ যাতে প্রযোবিক copy রাখে, এবং Magnetic tape ও hard disk রাখে।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

Cache

* ~~Fast~~ ^{Fast} Cache এ data check করতে। না পেলে RAM এ যাবে। Cache Fast

Direct Memory Access

* প্রতি 8 bit data transfer এর জন্য CPU দ্বা-
interrupt করা হয়।

* Data ও instruction সব main memory তে
মাঝেন A Von Neuman architecture

আগে প্রত্যেক operation এর পর CPU দ্বা-
interrupt করা হতো, next instruction Fetch
হবে। কিন্তু প্রতি byte কে একটি interrupt করা
অসম্ভব। তাই DMA দ্বা-কর্তৃ একটি block
ক্ষেত্রে CPU দ্বা-interrupt করা হয়। block হলো
bytes এর মিশ্রণ।

Sat Sun Mon Tue Wed Thu Fri

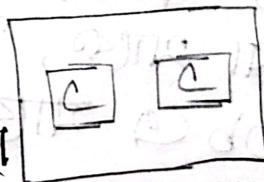
Date: / /

Sub:

Multi-processors:

* একটা chip এর ভেতরে ছাইটা CPU.
একই core বলি।

* এছালো parallel & tightly-couple মানে
যুগ্ম closed and communication খুব
Fast হয়।



* OS খুব সহজ switch করে একটা
'থেকে আড়েন্ট' বাতি করবে, একটা CPU থাকলে বাতি
করি one at a time একটাই করবে, তাই core যেকোন
ইলে দান দিচ্ছি হবে, but ফিল মনস্যাও আছে
multi-processor. Multiple CPU থাকলে speed
performance অনেক হও এমন নাই।

* আলাদা chip ছাইটা CPU এর মাঝে থাকলে যেটা
apply করা যাবে।

Fault tolerance → অনেক Fault হলেও performance
একেক করবেনা, tolerate
করে নিবে।

আলাদা আলাদা core থাকবে আলাদা আলাদা
task এর জন্য → এখন more popular.

Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

Multiprogramming :-

- * আগে memory তে স্থান নিয়ে রাখা ক্ষমতা
ক্ষমতা সেটা load করা, কিন্তু এখনে
computer Idle থাকবে।
- * তাই এখন একটা way আছলো যে memory
তে একটা একটা করে multiple job load
করা মাত্র, যখন I/O আসবে তখন আরেকটা
job এ মাঝে এচা multiprogramming.

* CPU job করতে করতে কোনো switch ক্ষমতা,

Multitasking / time sharing:-

- * Multiple job আছে বিন্তু এজ frequently
switch করে যেন delay হুক্ম না আয়,
job ছালোর জান্মে swapping হত একবে।
- * job ছালোর জান্মে swapping
of processes ক্ষমতা

Sub: _____

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

Date: / /

Operating-system Operation:

Interrupt Driven

↓
Hardware

↓
Mode bit

↓
CPU execution এর

instruction mode switch

করতে পারি।

↓
Software

① Infinite loop এ

প্রতিটাই

② কোনো error

হলেও

* privileged instruction only kernel এ করা যায়।

System call → OS যাত্রে privileged support
চাই, এটি user আর Kernel এর
সাথে

যোথ্য memory allocate করবে } kernel
করতে পারে ন ন করবে } mode.

* trap এর ফল mode bit switch হবে।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

* Dual mode use করা হয়। User mode
ধূর যেকী privacy রয়েন। আবার, user mode
এসব operation হ্যান অনন্ত file opening,
তখন support হিসেবে kernel mode call
করা হয়, যা কিছু privileged পাই। এজন্য
system call করা হয়।

APL API

Windows API

Linux

fid from

APL routines via
Native share routine

API APL

File I/O & Kernel API via routine

kernel { IPOS galloc sysman ioctl
· share IPOS N N

· IPOS native fid share map AC about

✓ Sat Sun Mon Tue Wed Thu Fri

Date : 23/09/2023

Sub:

Computer System Architecture

Operating system handle করে:

- * processing
- * Memory protection
- * Memory allocation
- * storage handling
- * Mapping

* storage কে OS manage করে File system
এর মাধ্যমে

Sub: _____

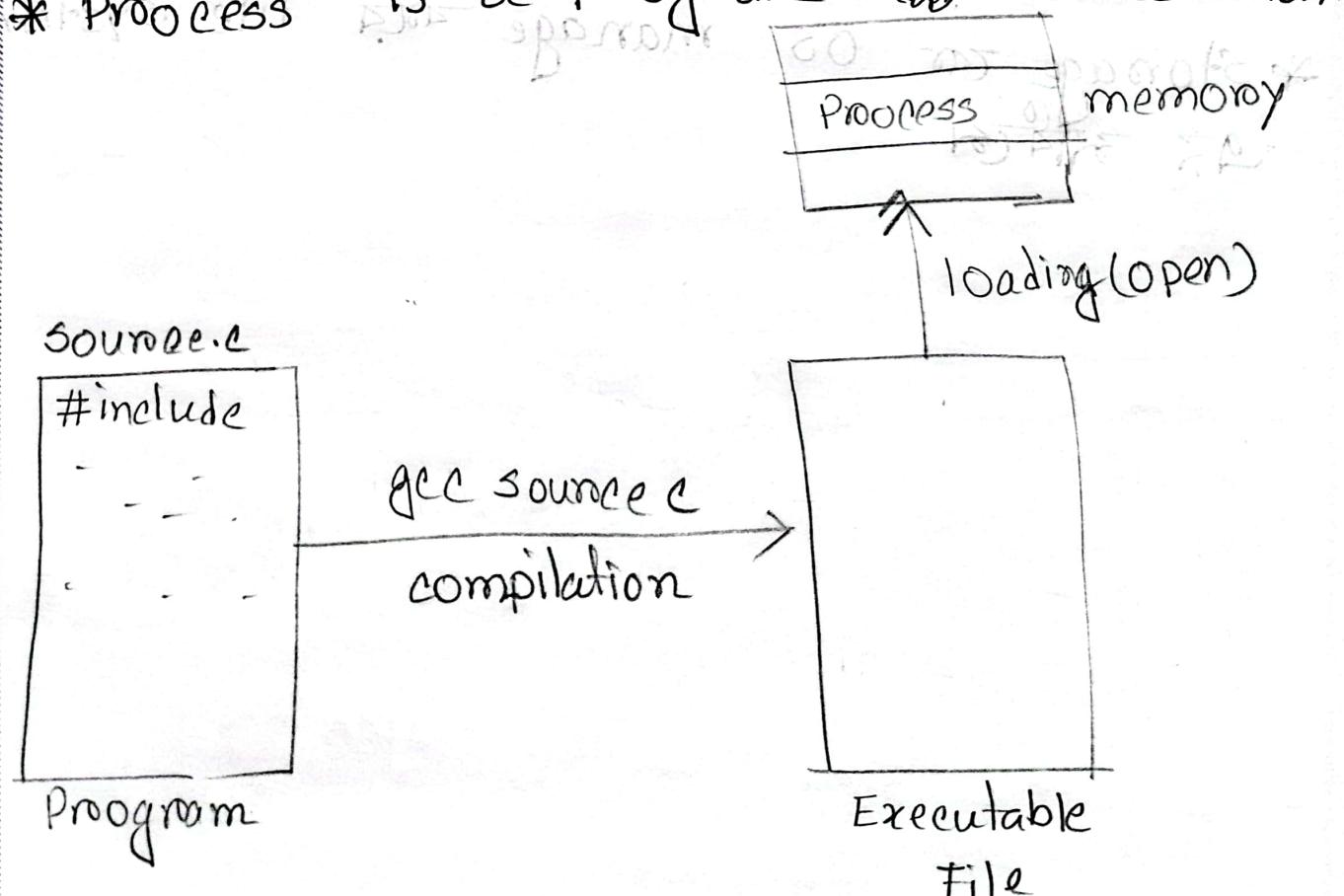
Sat Sun Mon Tue Wed Thu Fri

Date: / /

Chaptero-3

Process

- * OS ultimately process execute करते।
- * Batch system / Multiple system (गाडी) job share करते।
- * Time - Shared system → user program or task.
- * Process is a program था in execution.



Sub:

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

Date: / /

* program counter works like IP

Stack এ local variable রাখার কী ফুর্কি :

* particular function এর local variable
গুলো operation এর পর আবার ফিরে
পাই,

* program passive entity তাই সেটা disk এ store
যাবে, তাকে অন্য যথন load করে run করব
তখন active হয়,
→ program

* single file থেকে multiple process create
হতে পাও।

* text, data এদের storage fix.

* stack ও Heap dynamically allocate হয়।

* stack এর growth downward

* stack এর growth upward

* stack, heap memory share করে।

Sat Sun Mon Tue Wed Thu Fri
Date: / /

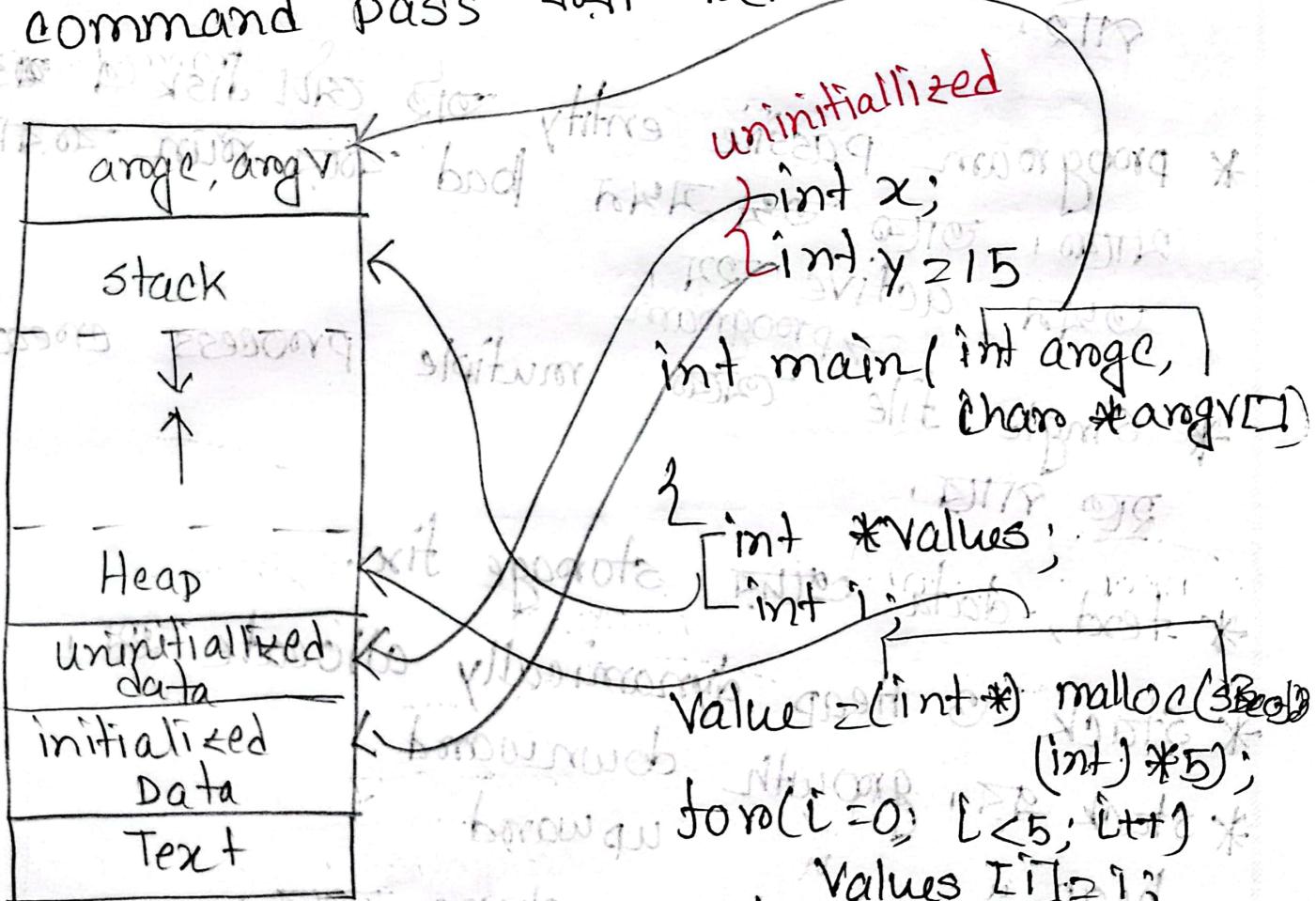
Sub:

`int main(int argc, char *argv[])`

Argc → Main function ও command টি জো

* main function টি call করার প্রকার যথে

command pass এব্যায় যাবে



* malloc টির দ্বয় data access করে তা থাকে
heap

Sub: _____

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	

Date: / /

* Heap ও stack dynamically shrink or grow করবে।

* প্রত্যেক process এর নিখিল stack ও heap আছে।

প্রত্যেক process এর state কয়েকটি রূপ আছে।
কাউন্ট করলে ৫টি।

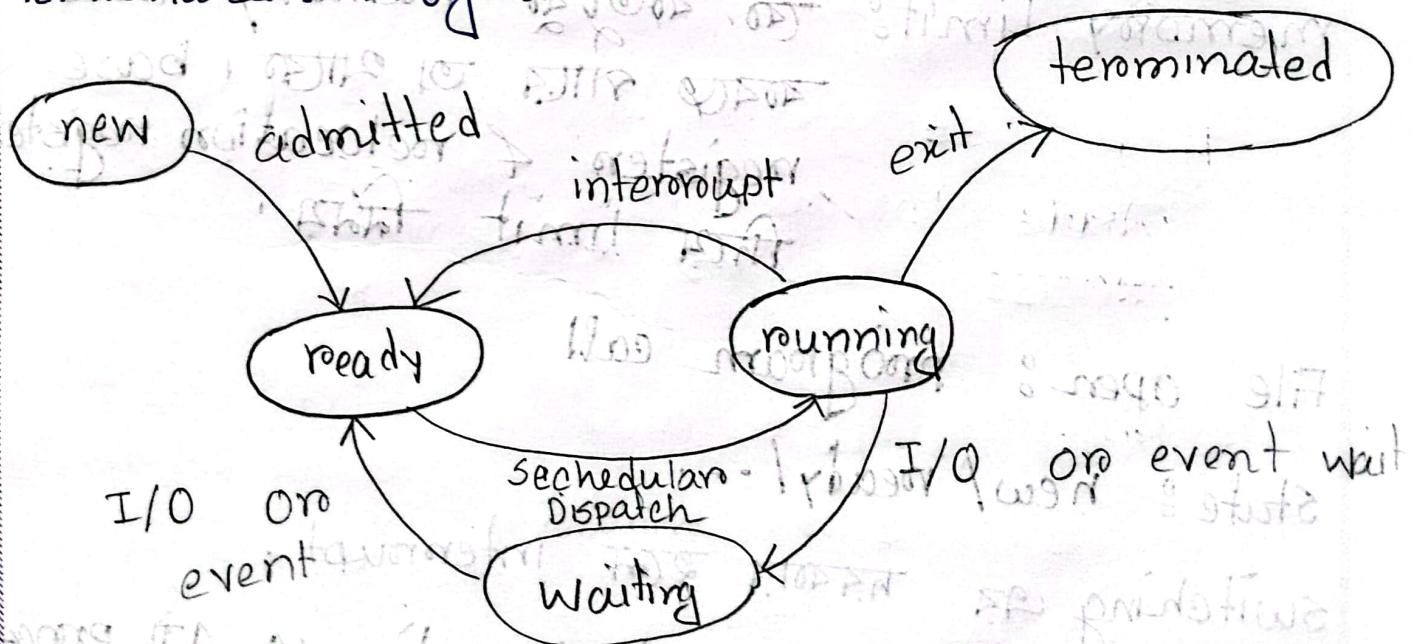
New : Process create করছে।

Ready : Process run করার জন্য ready but CPU কে আপ্স হয়েন।

Running : CPU কে run করছে।

Waiting : I/O এর জন্য wait করছে।

Terminated : Program terminate করছে।



* Interrupt করা ক্ষমতা suspend হয়ে যাবে।
ready-step এ যাব।

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Sub: _____

Date: / /

* event wait: সালে যেনের পার্টিকুলার এভি এবং
তার উপর কাজ করছে।

* প্রক্রিয়াক প্রক্রিয়া এবং সালে associated data
structure আছে যা process control block.

* প্রোসেস এবং বাইনে থাকে

program counter: Last দ্বাৰা পর্যন্ত execute
হয়েছিলো তাৰ অন্তর্ভুক্ত information
যেন context switching কো
support কৰতে পাবে,

memory limit: কো কান্ট্ৰু মেমোৰি access
কৰতে পাবে তা থাকা : base
register & relocation register
ধৰ্মী limit নিয়ন্ত্ৰণ।

File open: program call

state: new / ready / --

switching এবং দ্বাৰা হলে interrupt.
switching এবং দ্বাৰা হলে interrupt.

context switching: Particular time Δt process
কৰক অন্তর্ভুক্ত প্রক্রিয়া এবং switch.

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: 30 / 09 / 2023

- * Process নিতেও একটা data . তার সঙ্গে আরো data store করতে।
- * যেনন ছবি নিজ একটা data আবার ছবির কিন্তু data থাকে, যেনন কখন তোমা, তোম এসব।
- * Process data নির্দিষ্ট address এ store থাকে।
- * CPU র parallelism না থাকলে জানে একটা process at a time বললে তা desirable না, তাই একটা process execute করতে সাবেক্ষণ্ট switch করা হত।
switch করে যে process র জাত যৌগ করে আবার আজোর process র back করলে যথাজ আবার আজোর process র মেঘান প্রেতে শুরু করতে প্রয়ুন switch করবেছিলো। যেমন একটা করতে প্রয়ুন একটা data store করা লাগতে, address মেঘে না। এজন্তু data store করতে লাগতে।
- * একটা process এর address অন্তে process access করতে পারবেন।

Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

Process block এবং ফিলিটা:

আছে

- * P_0 executing phase এ আছে, তখন তাকে interrupt করতে তাহলে P_0 মন্তব্য হয়েছে তা process control block এ save হবে এবং process P_1 execute হবে, আবার P_1 আবেষ্ট করে P_1 save করে P_0 . এটাকে P_0 interrupt করলে P_1 save করে P_0 execute করবে।

Threads:

- * single threaded program নামে দাও at a time একটা execution stream এ আছে।
- * Process এর child process করা হয়। যেমনঃ chrome এতো, process এর child process graphics rendering.

- * Multithread এ data, file সময় তো সেই register, stack আলাদা, code share করতে thread আলাদা, নামে আলাদা thread কিন্তু ফিলিটা বাই করতে।

Sub:

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

Date: / /

* ~~একটি processor এর under~~ multiple threads run করানো তা data, files share করবে, code share করবে। একটি thread করবে।

* ~~কিন্তু~~ multiple processors এ আলাইt আলাইt child process run করে, এরা process share করতো।

* Thread এ process share হবে যখন data sharing easy. child process এর দ্বারা multi threading করা easy. Time save হবে।

* Thread switch করা fully parallel হয়না; fully parallel hardware support লাগবে।

→ it এ আছে, actual CPU র core count মত core তা
* Hyper threading এ প্রত্যেক processor এর under
threads মাঝে।

SS	SS
SS	SS

Parent - child relationship: Linked list

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
0	0	0	0	0	0	0	0

Sub:

Date: / /

Time-slice: প্রক্রিয়াক প্রক্রিয়া নির্দিষ্ট time এ

কোম্প না হলে switch করবে।

Time sharing switch: 1 টা process এর জন্য নির্দিষ্ট Time বেঁধে কাজ করা using time slice

Process scheduling:

* একটা particular time এ CPU তে যেকোনো process run করবে।

* CPU ক্ষেত্রে অন্য পক্ষে না থাকে।

Process টলতে থাকে।

1. Job Queue: Job hard drive এ আছে memory তে execute.

2. Ready " : Memory এ আছে execute

3. Device " : I/O এর জন্য ready wait
করতে execute করার জন্য,

Terminal unit \rightarrow display

disk \rightarrow internal hard disk or blind - free disk

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

Schedulers : OS এর খুবই core component Kernel Level এর ক্ষেত্রে job queue তে মাঝে।

qui → core component না

→ n n হলো scheduler, memory management

Magnetic tape → খুবই সমস্তা, huge space, achieve এর Level এর

* ready queue থেকে CPU তে মাঝে, terminate ক্ষেত্রে অবাধ I/O request কে মত পাওয়ে, এবং depend ক্ষেত্রে scheduling algorithm এর উপর ভেঙ্গে রাখে।

Forok a child! Parent & child same হবে।

copy version হবে, ubuntu কে fork কে modification এর জায়গাতে Lubuntu তেরি করা।

* parent execution এ ছিলো child এর জন্য ছিলে পিছে waiting এ মাঝে,

* জানে জানে ক্ষেত্রে process interrupt এর জন্য wait করে, যতক্ষণ interrupt হবে wait করে interrupt হলে ready queue তে মাঝে।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

* Loading separate thread (এ না হলে আস) একই thread এর effect রয়েলে, যাতে hang হওয়া
শাব্দ process রয়েলে

Multi-thread application \rightarrow I/O request আমতি
thread response
আমতি thread.

Q: Difference between process and threads

Sat Sun Mon Tue Wed Thu Fri

Sub:-

Date: 01/10/2023

Schedulers:

3 types. But 2 types are important

1. short term scheduler: process ready queue থেকে CPU টি নিয়ে আবে।

2. Long term scheduler: কোন কোন process এর disk থেকে ready queue থেকে নিয়ে আবে।

* I/O bound process: I/O operation এ বেশি time রয়েছে, এই process database একে use করে,

* CPU bound process: Execution একে, CPU র বেজ একে computation একে রয়েছে।

* memory গত জায়গা বর্জন হলে RAM এ ready queue গত process রাখা হয়।
process এর address কে ছোট ছোট পার্ট করা → paging

Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub: ECE 0110

3. Medium term scheduler:

Degree of multi programming

মাত্রা অন্তর্ভুক্ত প্রক্রিয়া এবং মেমোরি টেক্সুল প্রক্রিয়া

load করা হয়।

* Process টেক্সুল execution এবং swap

বর্তে RAM এ রাখে।

Multitasking in mobile systems

* iOS এ background process এর interaction

করা হাতে আর suspend হয়।

* iOS এ নতুন process শুরু করা হচ্ছে এবং

interaction হচ্ছে এমনভাবে যাকি অব

বন্ধ করে দেওয়া হয়।

* foreground এ process চলতে হাতে, background

এ off হয়ে যাবে।

* Android এ service API এর ক্ষেত্রে long

running process হিসেবে background এর run

হচ্ছে। কিন্তু করে process.

Sub:

Sat Sun Mon Tue Wed Thu Fri

Date: / /

Context switch :

- * এক process থেকে আবেক process এ switch করানো file download ফিল অন্ত process এ switch করা।
- * context switch না থাকলে আর download ফিল word এ বাতি করতে পারতানা।
- * switching খুবই fast হয়।
- * Process এর idle স্টেট করানো যাব।
- * single core, single thread computer মাঝে switching possible না।
- * switching fast না হলে hang করাব।

Process creation :

- * আজোয়াজ process ই run করায় তা child process
- * আজোয়াজ process open করি তাৰ parent process থাকতে পাই,
- * child process আজোয়াজ process create কৰে child tree তৈরি কৰে,

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

* Parent and child concurrently execute

বাবতে পারে।

* Parent process sometimes child process
create করে specific ফল result এর জন্য।

ISLAM: আর তাই wait করে child process করে
বৃক্ষ কলা।

* Parent child resource share বাবতেও পারে,
আবার child নিতুর কাজে use করতে পারে

only

Multi process Architecture → chrome এ

Tab চালু করা।

Inter Process Communication:

* দুই process এর মধ্যে communication

1. Independent: যেনে process এর effect

পরিনা,

2. Cooperating: অন্য process এর effect পার।
IPC mechanism দ্বারা।

Sub:-

Sat Sun Mon Tue Wed Thu Fri
Date: / /

* একটা process এর output আবেক্ষণিক
input হিসেবে কাজ করবে।

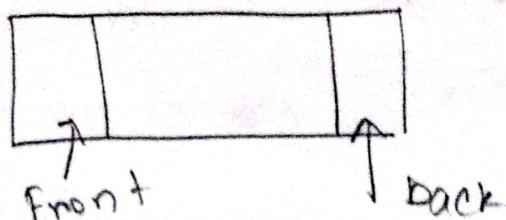
① shared memory : memory share করে যোনো
process execution করবে।
communication process এর
উপর depend করে, CPU'র
যোনো control নাই।

② Message passing : একটা process আবেক্ষণিক
message পাচিনোয় সাধ্যতা
communicate করবে,
CPU'র control এ
সহজে, Time দ্বারা লাগে
এখানে,

Producer Consumer Problem : (shared memory paradigm)

* Producers create করে shared buffers এ
বাধ্য consumer পরে তা use করে।
Circular buffers.

queue:
(circular)



✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

in pointer প্রযুক্তির মধ্যে বিভিন্ন সেন্ট
করবে।

out এর মধ্যে queue এর front.. consume এর

মাধ্যমে করা হয়।

at a time (buffer size - 1) data size এর

item রাখা যাবে,

$$\boxed{\text{in} = \text{out}}$$

খালি, buffer এ কোনো data নাই,

* Producers & consumers একই common buffer
share করে, একটিকে produce আর আরেকটিকে
consume করতে

queue এর front \rightarrow pop operation

while ((in+1) % buffer-size) == out) \rightarrow true
হলে full.

\hookrightarrow check in এর পরে next দ্বর কোনো কিনা

✓ Sat Sun Mon Tue Wed Thu Fri

Date: 04/10/2023

Sub:

short

- * Bounded-Buffer shared memory টির implement
কৰা যায়।
- * shared memory ও message passing এর
differences
- * message passing এ os কে request কৰতে OS
call কৰে message pass কৰবে
- * message passing → queue system: আলাদা PC তে
message passing 2 operation এর জাগ্রুতি হবে
- * Web browsing পুরো system → message passing
Message passing implementation 2 রকমঃ
 - ① Physical, ② Logical.
- Over network system → msg passing → Loosing
bounds
- * shared memory এবং computers এ Feasible
but Network এর concept এ Feasible না।

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

Direct communication!

- * Process explicitly address বরবার করে।

- * Ambiguity না হয়।

Indirect communication!

- * Sender receiver দেখ direct করতে পায়।

- * মানদণ্ড করা হয়।
refers করবার জন্য একটি link
mail box share করবার জায়গে link
establish করা হয়।

- * multiple mailbox exist করতে পায়।
যা same টি ভিত্তি share করতে পায়।

- * symmetric addressing → same sender & receiver

- * multiple readers, writers এবং ক্ষেত্র mailbox
exist করে।

- * Receiver always fixed. Receiver mailbox
select করব।

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

- * at a time একটা process কে পেরোতে পারে।
মন্তব্য করা হবে।

Message Passing → blocking → synchronous
receive

- * Sender কে block করা হবে। মতুজন পর্যন্ত send না হয় completely.
- * Receiver কে block করা হবে যতুজন না available রাখে data message.

Message Passing → Non blocking → asynchronous

- * sender + receiver blocking গ্রহণ করে। বাধন message buffer full হওয়ার chance নাই। বাধন consume ও pass না হওয়া পর্যন্ত ক্ষেত্রে কাউকে করবে, block হয়ে থাকবে।

- * shared memory টে security issue হবে।

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

Buffering Capacity:

মাল্টি link দিয়ে pass হবে তার capacity ক্ষেত্র।

① zero bounded → message queue

bounded → N মেসেজ স্টোর করবে।

socket:

IP address shared / public হবে।

* IP unique না হলে socket operations

থানা

* port Number + IP address combination
নিজে process করে এবং বেছে নেতৃত্ব করবে।

বরবা যাব।

port Number + IP Address → Socket.

* socket to socket communication হব।

* Web servers → basically software

Sub: _____

Sat Sun Mon Tue Wed Thu Fri

Date: / /

* servers হলো certain port এ run কৰে।

Maria DB

Web server

SSH server(22)

FTP server(21)

HTTP server(80)

port

→ secured shell → অন্য computer এ
login কৰা যাবে

* port medium process connect কৰার জন্য।

* web browser client মা communicate কৰবে
web server এর মাধ্যে, web servers এ port
map কৰা যাবার ফলে সেটা hit কৰবে এবং
show কৰবে টি port এ কোন process running
তাৰ মাধ্যে।

localhost 127.0.0.1 → নিজেৰ computer'ই।

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

pipe → for taking output from one command & use it as input for another command

command 1 | command 2

Types:

Unidirectional

Bidirectional

Unidirectional

(S)ource side

(L)evel side left

(R)everse side right

→ distinguished between L.H.S & R.H.S
R.H.S → output

→ for taking output from one command & use it as input for another command

→ distinguishes the levels as upward down

foras → reverse down, while PR → reverse down

→ PR → P.D. foras → P.D. PR → Q.P.M.

→ reverse 2236.0079 foras → foras → P.D. work

→ P.D. → A.G.

→ distinguishes output ← I.O.C.F.LI → final level

Sub:

Sat	Sun	Mon	Tue	Wed	Thu	Fri
0	0	0	0	0	0	0

Date: 07/10/2023

~~UNIX~~ UNIX এর configuration powershell এর দেখ-
অনেক better option.

Chapter-6

CPU Scheduling

Objectives:

* Ready queue যে process আছি তা certain condition match করলে তা CPU তে দাওয়া

হো।

* Basically schedule করি বুরা।

* কিছু কাজের জন্য CPU এ অবস্থা দিএ, কিছু সময় I/O তে দিএ।

* এই যে একবার CPU জ্বাবার I/O এটা হলো-
Burst.

* Burst মানে অল্পমত্তা একটা কাজ করে আবার switch করবে খুব কম।

* lower burst duration এ frequency
অনেক বেশি than higher duration.
process হলো এমন যন্তা খুব long time
ধীরে use করবে।

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

Date: / /

Sub:

* interrupt ছাড়াও switch করতে পারে।

* Burst duration খুব কম CPU র মানে

কখনো process এর CPU টে খুব একটা

time লাগে না, তাহলে CPU unutilized

হবে, কিন্তু CPU কে unutilized

করা যাবে। এই ক্ষেত্রে multi programming

করতে হবে, তাহলে CPU burst ক্ষেত্রে

I/O burst করুক হলে, CPU টে তখন

memory টে loaded process কে তথ্য আ

দিতে হবে।

বাধন CPU Scheduling হবে।

* Running থেকে waiting stage এ হলে

CPU হাত দে হলো, এধন processing কে-

new schedule দাওয়া হলো,

* Running থেকে ready state এ হলে

process schedule করা যাবে।

* Waiting থেকে ready state

* Terminate, করলে নতুন process

start করতে পারে, কিন্তু কোভা

Sub:

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

nonpreemptive: Running # যখেন তাকে সরানো
মাধ্যম।

preemptive: Process কে সরানো মাধ্যম।

* running process কে ready টে নিয়ে মাধ্যম
সাবেক তাকে stop করানো মাধ্যম preemtive.

Dispatcher:

Dispatcher responsible for
schedule কে processors এ অন্তর্ভুক্ত
দেওয়ার ক্ষেত্র।

Dispatch latency → একটি process কাউন্ট
করবে আরেকটি process
start করার মাধ্যমে এ
সময়।

* Ideally টাই টে schedule এর time কম লাগবে।

✓ Sat Sun Mon Tue Wed Thu Fri

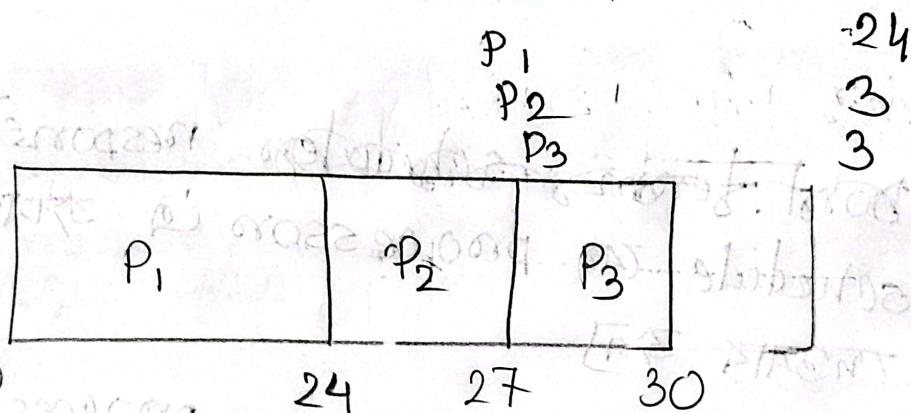
Sub:

Date: / /

Scheduling Algorithm:

First-come, first-served:

* Arrival time টেক্সা না মাজেন্সে এই sequence
এ আসতে এই sequence এ বিরুদ্ধ নিত হবে।
Process burst time



1st এ আসে P₁, P₁ এর মাঝে 24 মাটি
যাব্বা, এরপর 2nd, 3rd এসবে করতে হবে।

* response time 0. মানে বাতাশুনের সম্ভাৰ
answer দেওয়া কৃত কৰিব।

P₁ = 0 → মাঝে মাঝে executing কৰাই। response
time 0

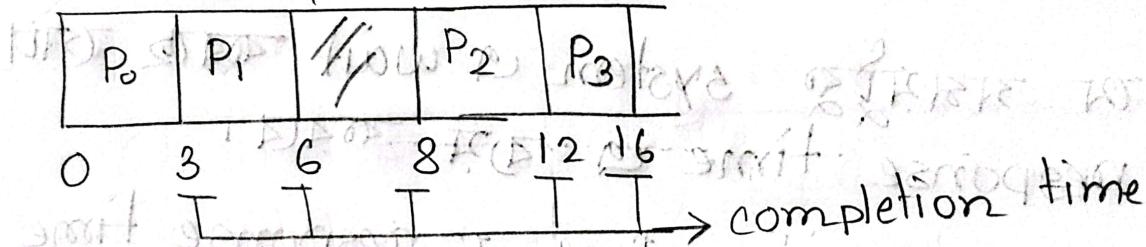
P₂ = 0 + 24 → + 3 → burst time

$$P_3 = 27 + 3 \quad \text{Avg waiting time} = \frac{0+24+27}{3} \\ = 17.$$

Turn around time \rightarrow System \rightarrow କାମିତେ time ଲାଗୁ

First come first serve এখনই best solution নিবে
যখন হোক process আগে, বড় process পরে থাকবে,

P: 6.12



turnaround time = $\text{execution time} + \text{waiting time}$

TM system ক্ষতি time যুক্ত করছে।

ideal case এ ~~চাই~~ process system-এ burst time
এর মজাত time spend করবো, but আর সাথে
extra factors হিসেবে যাই এই waiting time যা-
undesirable.

Sub:

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Date: / /

$$\text{Turn-around time} = \text{Completion time} - \text{Arrival time}$$

$$\text{Turn-around time} = \text{Burst time} + \text{Waiting time}$$

$$\Rightarrow \text{Waiting time} = \text{turn-around time} - \text{Burst time}$$

* যে মন্তব্য করা হলো যে প্রতিটি process এর response time এর সময় অনেক কম।

$$\text{process } \quad \text{Waiting time} = \text{response time}$$

Q. Preemptive system এ waiting time \neq response time but why?

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

* arrival time দেওয়া না থাকলে রিঃ নিয়ো।
যের packet same time এ arrival করবে।

P_4	P_1	P_3	P_2
0	3	9	16

P_1	P_2	P_3	P_4
0	24	27	30

* avg. waiting time সবচেয়ে কম আসব।

• shortest job first scheduling when arrival time is different:

* non-preemptive এ কাউকে জায়গা দিলে তাঁর না
হওয়া প্রয়োজন।

* arrival time দেখা লাগবে, এ process available
কিনা দেখতে।

Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

* System এর overall turn around time
কোনো মাঝ কিনা।

$$TA = WT + BT$$

$$WT = TA - BT$$

* একবারে CPU থেকে তাকে অবানে যাবেন।

* decision making এ burst time এর value
নেই। Arrival time পর্যন্ত দেখছি।

* Burst time to assume করা যাবেন।
তাহলে shortest দ্বা কীভাবে select
করবে।

* Assumption পর্যন্ত হবে statistical method
যিনি অতীত কী হয়েছে তার উপর
base করে।

* exponential moving avg যিনি 75 burst
হয়ে গেছে তা যেজো যের কুরতে হবে।

* $0 \leq \alpha \leq 1 \rightarrow$ moving factor

* Recurrence Relation কে expand কুরার
সার্বিজে যের কো যায়।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

$$T_{n+1} = \alpha t_n + (1-\alpha) T_n$$

ক্লেন ফরম করতে weight দিয়ে depends
on α .
 $t_n \rightarrow$ actual burst
 $T_n \rightarrow$ guess burst
 $\alpha = 0$ হলে

$$T_{n+1} = T_n \quad \rightarrow \quad T_{n+1} = t_n$$

$\alpha = 1$ হলে actual CPU burst count হবে,
usually $\alpha = 1/2$

$$T_n = \alpha t_{n-1} + (1-\alpha) T_{n-1}$$

এর জন্ম এবং
করতে আবার-

$$T_{n+1} = \alpha t_n + (1-\alpha) \alpha t_{n-1} + \dots$$

$$+ (1-\alpha)^j \alpha t_{n-j} + \dots$$

$$+ (1-\alpha)^{n+1} T_0 \quad T_0 \rightarrow \text{initial.}$$

$$\sum_{j=0}^n (1-\alpha)^j t_{n-j}$$

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

* Ist এবং দ্বিতীয় term higher orders এ রাখে
আর মতো যতো কম রাজত মানবে।

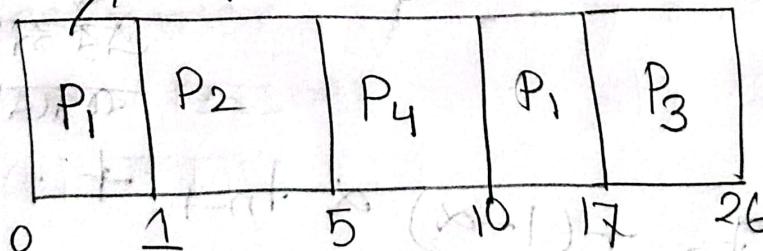
* এসেও সোচাজোটি আলো একটা prediction
পিএইচ ফর নেক্স এপি ব্রাউন্স্ট।

shortest remaining time first:

* * preemptive algorithm.

* কোন process এর remaining time

কম সেটি execute করবে।
originally করবেনি।
→ 1 burst



Remaining
$P_1 \rightarrow 7$
$P_2 \rightarrow 4$
$P_3 \rightarrow 9$
$P_4 \rightarrow 5$

P_1 এর চেয়ে P_2 এর remaining time কম
তাই P_2 এর priority কমিগ, তাই P_1 কে
মরানো হবে, P_2 execute করবে

P_3, P_4 arrive করলেও P_2 এর remaining time
সবচেয়ে কম।

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

এক্সা process complete হলে পরবর্তী শুলো remaining time check করে আবার ব্যবহার হবে। same process এই হবে।

Waiting time,

$$P_1 \rightarrow 10 - 1 = 9 \text{ msec}$$

~~$P_2 \rightarrow 1 - 1 = 0 \text{ msec}$~~

$$P_3 \rightarrow 17 - 2 = 15 \text{ msec}$$

~~$P_4 \rightarrow 5 - 3 = 2 \text{ msec}$~~

nonpreemptive \rightarrow nonpreemptive
 $\text{response time} = \text{Waiting time}$

Wait করে

For fast

response

$\text{response time} \neq \text{Waiting time} \rightarrow$ preemptive

*non-preemptive 1st waiting \rightarrow response time.

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

Response

$$P_1 = 0$$

$$P_2 = 0$$

$$P_3 = 17 - 12 = 5$$

$$P_4 = 5 - 2 = 3$$

* process এর state diagram → চিন্তা হবে।

Priority Scheduling:

* সব process এর priority অন্তর নাই।

* OS এ Kernel আৰু user mode এৰ process এর priority same নাই।

* যে process এর priority বেশি তাকে আগে দিত হবে

* non preemptive →

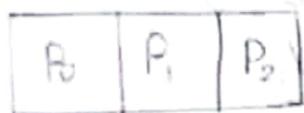
* preemptive → higher priority process আবাসনে lower পৰি আয়ত্ত switch

Round Robin Scheduling:

* Response time অবচেম্প lowest, তবে not necessarily waiting time কম

$$q = 10 \text{ ms}$$

RQ



↳ Ready Queue

P ₀	P ₁	P ₂	P ₀	P ₁
10ms	10ms	10ms	10ms	10ms

RRS এ নিমিষ অন্তর q বাধা।
নিমিষ অন্তর process ক্ষেত্রে হলে context switch হয়ে আবার execute হবে।

Why response time অবচেম্প lowest?

n → number of process in ready

at most $(n-1)q$

যদি গোলা process time quantum এর আর্টিক-ক্ষেত্রে যাই তাহলে $(n-1)q$ পর্যন্ত অপারেটিং সিস্টেম ক্ষেত্রে হবে।
আর- at most বলো।

q এর value অনেক বড় হলে Fast come, fast serve

q নুনু নুনু ছোট নুনু অনেক বড় অবচেম্প context

switching এ যাবে, execution এর চেয়ে বড় অবচেম্প context switching এ যাবে। তাই Not too small not too

large value of q.

Sub: _____

time quantum এর টেক্স ব্রুষ্ট টাইম হো
ক হলে, process কেবল হলেই switch.

Response time:

$$P_1 = 0$$

$$P_2 = 4$$

$$P_3 = 7$$

$$q = 4 \text{ ms}$$

$$\begin{aligned} &\text{at most} \\ &= (3-1) \times 4 \\ &= 8 \text{ ms} \end{aligned}$$

Waiting time:

$$\rightarrow (10-4) = 6 (\text{P}_1 \text{ এর জন্য})$$

$$\rightarrow 4$$

$$\rightarrow 7$$

waiting time অবচেতনা SJF (Shortest Job Finding)

Why Round Robin?

→ অবচেতনা response time বজায়

→ waiting time সাহাজেরি মেশিন না

→ অবচেতনা equally execution এর মধ্যে সম্মত পদ্ধতি
সব process এর execution সম্মত পাবে।

q এর value অনেক বড় হলে context switch হয়ে
বা অনেক বজায়।

q এর value অনেক ছোট হলে, context switch
অনেক ঘটে। আরম্ভাজন হলে balance হবে।

q এর value কিন্তু হলে \rightarrow FCFS, RRS সম্মত পাবেন।

Sub:

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

Date: / /

$$\# P_1 = 18$$

$$q = 20$$

$$P_2 = 1$$

$$P_3 = 1$$

$$n = 10$$

$$n\% = 80\% = 8 \text{ process} \rightarrow \text{optimal}$$

→ q

এরকম হলে average turn around time স্থানীয় পাও।

always time quantum এর value বাড়লে এখ best result পাও আ না।

→ optimal রাখা হব।

80% of the CPU burst should be shorter than q

→ Rule of Thumb

Rule of Thumb → সেরামতি optimal Result হব।

OS → hybrid system ব্যবহার করে, simple করে করেন।

Multilevel Queue: Ready queue তে এই process আছে, অবাধ জন্য algo apply করা যায়না, বিভিন্ন process এর জন্য বিভিন্ন scheduling algo. apply এর context

foreground → user এর মাধ্যমে interact (UI) → RR

background → আধে আছে result দরবার হচ্ছে → FCFS

2) scheduling → ① কোন queue select
 ② queue এর অন্তর্ভুক্ত process এর কোন scheduling algo

RR) foreground → RR → $q = 10$
 $q=10$) background → FCFS

এসব quantum এর value বিশ রাখায় হয় এসব চেয়ে

Multilevel queue টো → process গুলো queue switch
 করতে পারেন।

Multilevel Feedback queue: multiple queue মধ্যে
 মাঝে এবং process গুলো queue switch করতে
 পারে, এবং demotion / promotion হয়ে queue
 switch filtering কোম্প হিসেবে।

মুক্ত interactive process হলে 1st queue টো
 execute হয়ে যের হবে।

একটু বজ্জ interactive process হলে 2nd
 queue টো যাবে then complete না হলে
 FCFS.

Sub _____

process \rightarrow priority value (80)

\downarrow
change এখনা static

Nice value dynamically priority ছিল

-20 to +19, priority র সাথে তুল

Nice value বিমান বরতে হবে।

Numeric priority \rightarrow (0 - 140)

linux এ অরামদি multilevel apply করেন।

Time slice \rightarrow variable

- lower priority
- Higher priority

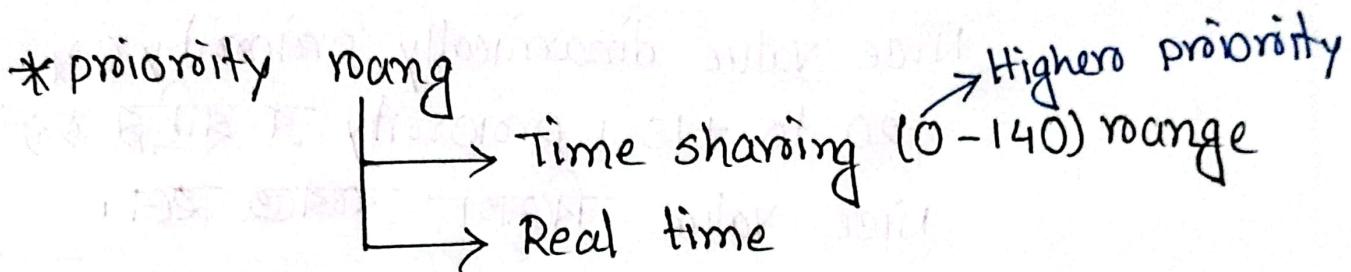
Sub:-

Sat Sun Mon Tue Wed Thu Fri

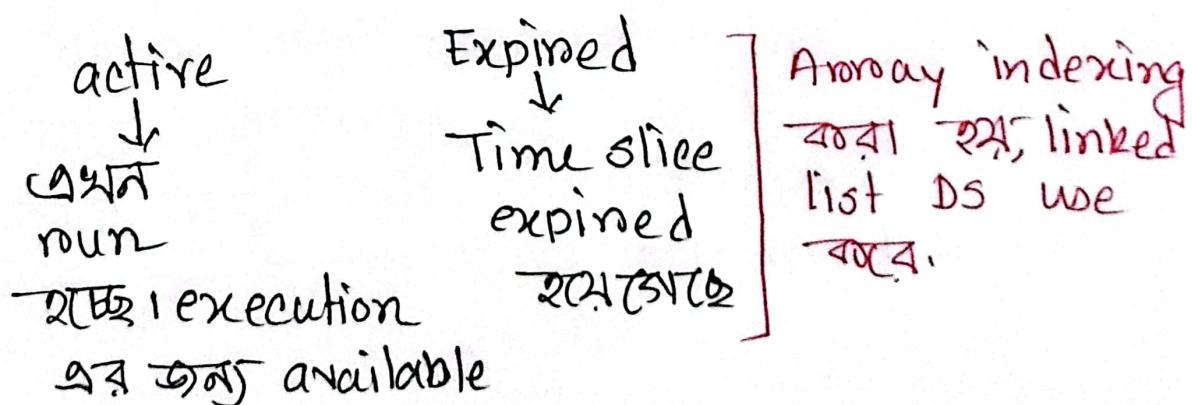
Date: 15 / 10 / 2023

Linux Scheduling

- * preemptive priority scheduling ব্যবহৃত 2-5 পদ্ধতি।
* priority wise সাজানো হবে।



- * Higher priority task এর lower time quantum তার, Lower priority task এর higher time quantum।
windows বা others. এসব reason হলো utilization বৃদ্ধি।
- * Linux এ এটা reversed. Linux কুড়ি array maintain করবে।



- * active array পরে expired হওয়া যাবে, swap করব।
আবার expired হওয়া জানে time slice এ কোথা হবে।
কোথা কোথা আবার active হবে।

Sub: _____

process \rightarrow priority value (80)

change রয়েনা static

Nice value dynamically priority

- priority (0-10) \rightarrow 20 to +19, i priority র বৃদ্ধির ক্ষেত্র

Nice value যিয়েন বরতে হবে।

Numeric priority \rightarrow (0 - 140)

Linux এ অরামধি multilevel apply করেন।

Time slice \rightarrow variable

\rightarrow lower priority

\rightarrow Higher priority

Chapter + 5

Synchronization

যদি multiple process same time এ shared buffer কে modify / write করতে চাহে, তাহলে synchronization এ problem রয়ে।

* 10¹⁰ element array টি গুরুত্বে value রাখতে পারাটি, 10¹⁰ value রাখতে চাইলে একটা আলাদা variable counters use করা হয়, যা buffers size determining counters use করা হয়ে, এর produce-consumers problem করবে, এর ফলে new algorithm ফিলে, Value store solve হবে, এটা নতুন algorithm মিলা check করবে।

* এখন counters variable producers and consumers টুটা share করে ফলে এটা সমস্যা create. করতে পাও, কী সমস্যা হবে এটা lower level operation থেকে জানা মাঝে, যদি shared variable write operation হয় সমস্যা তখন হয়, read এ হ্যনা operation হয় সমস্যা নেই।

* টুটা process concurrently run করবে, সাথে একজন produce করবে একজন consume করবে, টুটা linked হবে, producer কিন্তু time

Sub:

produce করবে, consumer বিন্দু time
consume করবে।

* এই অবস্থায় counters এর value চিহ্নিতে-
update হয়না, যখন concurrently run
হয়, low level এ মজব্বা হলো counter-এ
write operation delay হওয়ার কারণেই
register এ কোন value হচ্ছে।

* Race condition → যদি ২টা process same
time এ execute হবে, তার উপর base করে
কোনো variable এর value change হবে

* shared variable এর জন্য same time এ
write করতে দুয়ো পাই পাই থাইনা,

* loop ছাড়ি ইনে context switching এর
সুযোগ আসবেনা, বড় ইনে variable change
হতানা

এই problem এর solution process synchronization

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Sub:

Date: / /

critical section → Process শুল্কের particular section মাঝে তখন এ যদি shared zone এ enter করতে পারে, তখন অন্য কোনো process critical section execute করতে পারবেন।

3টি characteristics :

1. Mutual Exclusion:

At a time একটা variable কে modify করতে তখন এ variable fully modify না হওয়া পর্যন্ত অন্য process access করতে পারবেন। variable এ critical section এ থাকবে।

2. Progress:

যদি critical section → process execute না করে আর interested এ section execute করতে তাকে বাধিয়ে রাখা থাকেন।

Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

বিল্ড + Bounding Waiting: — waiting positions
নিম্ন time পর্যন্ত wait করতে ,
waiting time যো critical process
executing এর জন্য - নিম্ন time পর্যন্ত ,
time পর access করতে পারবেনা
এখন হওয়া মান।

* যখন A region এরা process execute
হবে mutually execution হবে

* একটা process critical section এ যেকে
অন্য process critical section execute
করতে পারেন।

Sat Sun Mon Tue Wed Thu Fri

Date: 04 / 11 / 2023

Critical section mutually execute করার algorithm

Peterson's Solution Algorithm:

* একটা process মাত্রেকাটার turn select করে নিয়ে।

Flag [] \rightarrow Ready to enter

Flag [] = True \rightarrow critical section execute
করতে চাই

Flag [] = False \rightarrow চাইনা

Turn \rightarrow whose turn it is to enter

* Peterson's solution দ্বারা process এর জন্য

Turn = j; \rightarrow Humble Algorithm

process i বলবে j এর পালা

-মাত্রা For j Turn = i জানে j বলবে i এর turn.

P_i
do { }

Flag [i] = true

turn = j

while(flag[i] && turn = j)

CS

পর্তা False

পর্তা True হলে পারবে

pi execute

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date:

flag [i] = false;

} while (true)

Pj

do {

flag [j] = true

turnn = i

while (flag [i] && turnn = i)

CS

true

true

50

loop

process

আর্থিক যাত্রা

flag [j] = false

ros

} while (true)

* P; ৩ Pj টুইটি chance পেলে P আরে execute

করবে।

flag [i] = true

flag [j] = false

turnn = j

Pi execute করবে কারণ

condition match করবেন

তব আঁকড়েন।

flag [i] = true

flag [j] = true

turnn = i

P, ৩ Pj টুইটি CS execute

করতে চাইবে but Pj এর

condition match করবে

হলে Pi loop এ আঁকড়েন
Pi execute করতে।

Sub:

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

Date: / /

flag [i] = true

flag [j] = true

~~turn~~ turn = j

→ এখানেও P_i & P_j CS execute করবে but flag [j] এর turn = j নিলে যাত্রাপথ $\rightarrow j$ loop এ আঠক যাবে। P_j execute করবে।

* Peterson's solution modern modern architecture
এ support করতে কোনো multi threading programming করা জন্ম

Sat Sun Mon Tue Wed Thu Fri
 Date: 12 / 11 / 2023

Sub: _____

- * computer optimization বৃক্ষাত পাবে।
- * Reorders এব় মার্কিজ , compiler বর্তে।
- * ~~each~~ Thread টুইটি variable share করে ও at a time টুইটি operation করে।
- * ~~each~~ Thread টুইটি concurrently run হচ্ছে।
- * ~~each~~ thread কোনো conclusion নেই এখন sequence maintain করা লাগবে Thread এ।
- * instruction reorders করতে পাবে এটাৰ output different আসবে, কাণ্ডে এই Thread এর value point হতা তা না হয়ে আবেক Thread এর value point হবে।
- * Peterson's solution একজনাথে ~~each~~ Thread critical section execute করতে পাবে, কিন্তু এটা mutual exclusion কে opposite করে।

Sub:-

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	

Date: / /

Process $i=0$

```

do {
    flag[i] = true; // switch
    turn = j;
    while (flag[j] && turn == j) {
        flag[i] = false; // CS
        if (turn == i) {
            turn = i;
            while (true);
        }
    }
}

```

Turn = 1
 $\{$ Flag[0] =
Flag[1] =
process 1 execute হবে
again after that

Turn = 0
Flag[0] = False
Flag[1] = true
process 1 CS
 \hookrightarrow আবাব CS
Then flag[0] = true
আবাব CS
এমাত্র process 2 এর
* 2 CS এ
কোন যাত্র at
a time time.

flag[j] = false; // CS

}

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

Sub: _____

Date: / /

Hardware based solution \rightarrow Atomic operation

Test-and-set lock:

locking mechanism

* variable lock আছে কিনা এটার উপর depend করে CS execute, করবে বাধন মেঠা decide করবে।

* একস্থানে execute করার সময় আর fully হবে নাহিনে হওয়াইনা

* lock acquire করে CS execute করে আবাব
an unlock করে দিয়ে, ফলে আবেক্ষণ্য Thread
lock করতে পারে, Means at a time একস্থানে
Thread এ CS execute করতে পারবে।

lock = 0 / unlocked

lock = 1 / locked

* lock 0 থাকলে তা 1 করে execute করবে
CS, আবাব as lock=1 থাকবে অধুন অন্তর্ভুক্ত
lock করতে পারবেনা ও একজনই execute
করবে।

conBasic Lock

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

Sub:

Date: / /

boolean test_and_set (boolean *target)

{

 boolean nov = *target;

 *target = TRUE;

 return;

}

2 Case possible \rightarrow b = False \rightarrow b = True

bool b = FALSE

test_and_set(&b);

↓

nov = False and *target = TRUE;

b এর value true

set করবে।

* যে value ফিরছে তা return করবে মানে b এর

value যা মানে true করবে।

* b = true হলে true point করবে ও b কে true

করবে।

* lock/unlock এ use করা হয় test_and_set

CS = Critical Section

Sub:

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

* lock এর value FALSE থাকলে loop এ
কাজ করবে তে এ যাবে, Then lock এর
value TRUE থাকবে।

* lock এর value TRUE হলে test-and-set
true return করবে ফলে CS execute
করতে পারবেনা।

* যে process CS execute করছিলো তার
execution এর পর lock এর value FALSE
করে দিয়ে, এখন অন্য process আবার
lock করে CS execute করতে পারবে

MUTEX lock:

* আলাদাগারে করা লাগবেনা। Function call
করেই করতে পারবে।

busy waiting → process এর আর কোনো
কাজ নাই, while loop
এ ছাঁট করবে,

Sub: _____

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

Date: / /

Semaphores

* Software based solution

* More sophisticated & general than peterson.

Variable:

{ wait(s) signal(s)
P(s) V(s)}

* If value neg or 0 then loop executes করবেনা
যদি ব্যাপে।

or

* signal এর কাজ $s++$ করা।

$s=1$

P_i

wait(s)

s2;

signal(s)

wait(s)

s1;

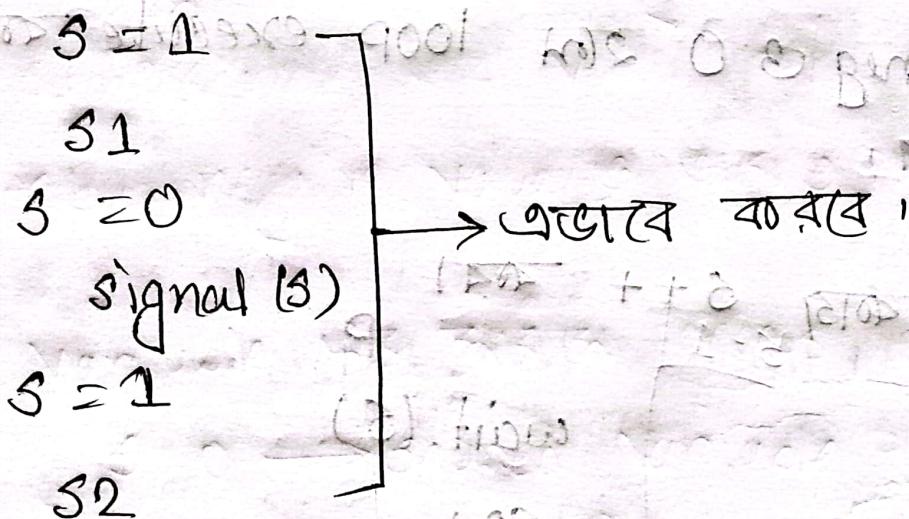
signal(s)

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

* P₀ আগে execute করলো তাই wait(S)
 call করলো: S এর value neg 0 & 0 হলে
 execute করতে পারবেনা 100P এ অটিক্ষম,
 যিন্তে S এর value 1 তাহে, করবে এরপর
 শেষস্থ ও ও এর value -1 হলে 0 হবে,
 এরপর signal(S), এ আবার 1 হবে, then
 P₁ same way execute করবে,



* সব মজায়, দুইটা wait(), ২টি signal()
 ধীকরণে এমন না, একটি wait(), একটি
 signal() থাকতে পারে, এমন হলে signal()
 100P এ অটিক্ষম যেতে পারে,

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
	○	○	○	○	○	○	○

Sub: _____

Date: / /

1. Counting Semaphore \rightarrow যাণোগো int.
2. Binary $n \rightarrow 0/1$

* Counting Semaphore \rightarrow multiple process
যাকনে use কৰা
হয়।

* আবু problem critical section problem না।

* Counting Semaphore = 5 জান তা- execute
কৰতে, বাবি আরো যাকনে possible না।
Mutual Exclusion এর আবু multiple
execution ও হয়।

* s এর value 0 এর চেয়ে বড় হলে while loop break
কৰতে।

* আর s=0 হলে wait(s) এর জন্ম যাবে execute
বস্বে এবং s এর value 0 কৰবে, এবং জন্ম অন্তর্ভুক্ত
হোল্ড কৰমে loop পড়ে যাবে, ক্ষেত্রে while loop
এর condition meet কৰতে পারবেনা, যখনে s execute
কৰতে পারবেনা, আবার s এর জান 1 হলে অন্তর্ভুক্ত
process কৰবে execute কৰতে,

Sat Sun Mon Tue Wed Thu Fri
 Date: 15 / 11 / 2023

Sub:-

* `wait()` call করে যদি কোনো process signal call না করে তাহলে deadlock situation এ পড়বে

`Wait()` → wait করতে হবে
 → resource available থাবলে execute করতে হবে,

* semaphore এর value 0 হলে screen lock এ থাবলে দয়া তাই negative value হ্যনা 0 or 1 হবে।

* But negative value ও আসতে পাও,

variable এর মাধ্যমে access control করা মাঝ-

`s20` → shared variable

Proc 1

`wait(s)`

31 execute → s1;

`signal(s)`

Proc 2

`wait(3) → while loop এ আসতে`
`32; (next spin lock)`

`signal(3)`

semaphore এর ক্ষেত্রে
`wait` এর বিপরীত
`signal(call)`

Sub:

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

Date: / /

wait()

critical section

signal

semaphore value $> 1 \rightarrow$ counting Semaphore

Wait Function call করে ২টা way :

① Actually wait ; যতক্ষণ রিসোৰ্স
available না থাকে

② আগে আগে execute.

* busy wait নি একে process sleep / suspend

* clock cycle নষ্ট হওনা

block \rightarrow ready queue to waiting queue.

context switch হবে only ready queue এর
process ক্লোর হয়।

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Sub: _____

Date: / /

Implement with no busy waiting.

এই semaphore এর জন্য মেরু-list, wait
করছে তাদের ১টি-link list এ store.

Process এর process control block (PCB)
link list store.

Semaphore S

S.value = -5

Semaphore এর value negative mean কান্সেপ্ট
পরিস্থিতি নির্দেশ করে।

Wait call

Semaphore S;

S.value = -1

S.link = {P₂}

S.value = -2

S.link = {P₂, P₃}

S.value = -3

S.link = {P₂, P₃, P₄}

resource available হলে আরো ৩rd process
semaphore এ wait-

Sub:-

Deadlock:

$$S=1 \quad Q=1$$

P₀ wait করেছে Q এর value কখন (+)ve হবে

P₁ " n s h n n (+)ve n

We have ইওয়ার জন্য signal call ব্যবহৃত হবে। but
signal call হওনা, spin lock-এ আটকে আছে,

Race Condition: execution sequence. এর নিম্ন দেখ.

P. (Semaphore execution টিক্কনতি implement না
wait (S) ইঁং,
wait (Q)

Then P_i FIFO OF Q_i has four IF L
wait(Q) } spin lock ready queue
wait(S) }

then P_0 ৰে যোৰত গিয়ে signal call হব, Then P_1 এ lock ছুড়ি আবে,

Blocked Buffer Problem

3 টা semaphore :

(at a time 1D process)

(i) mutex = 1 (binary) \rightarrow mutual execution
এর জন্য

(ii) full = 0

(iii) empty = n; } counting

\hookrightarrow initially buffer এ কয়েটা slot রয়েছে
আছে,

empty \rightarrow count, করে buffer এ কয়েটা অব্যুত্তা
যুক্ত আছে।

full \rightarrow buffer এ কয়েটা process থেকে producer,

mutex এর value 1 কর্তৃ 1 টি process.

1 বার wait call করলে ≥ 0 আর তোমা process
আপোতে পারছো।

wait(empty) \rightarrow buffer full থাবলে spin lock
full না থাবলে empty-1 off buffer এ 1D
process রাখবে।

wait(mutex) \rightarrow at a time একটি process

\hookrightarrow যদিওখোলা mutex = 1 না হবে process
buffer এ write করা যাবে না, (second
বার call)

Sub:

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

* consumer's side এ স্মার্ট buffer empty
তত্ত্বের block কার্য্য consume করার বিট্ট নয়।

* actually data প্রাপ্তি প্রক্ষেপ করে যাবে।
না করবে প্রাপ্তি এ spin lock.

consume → mutual exclusion way তে।

Q. Why 2nd semaphore variable?

⇒ n এর জান জানে n আগে ক্ষমতায়ে বিয়োগ করবে
আবেক্ষণ্য আবেক্ষণ্য value এ পাও, বিট্ট at a time
এ 2nd process বিয়োগ হতে পারে, mutual
exclusion way তে হচ্ছে কিনা check.

• 2nd semaphore ক্ষমতা সহ সেমাফোর ক্ষেত্র

• share semaphore ← producer স্লাইলিং

• stronger than the old one / share-based

• PBO - SCW

• semaphore extension

• PBO এর অধিকার ক্ষেত্র সহ সেমাফোর ক্ষেত্র

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: 19/11/2023

* Bounded-Buffer Problem with semaphore.



Readers-writers Problems :

* কুইটা process এ থেকে করা হয়েছে, read & write.

* কুইটা process একই সাথে read - write করতে synchronization problem দেখা যায়।

* এমন situation create করতে হবে, multiple men একসাথে read করতে পারবে, এ ক্ষেত্রে অন্তর্ভুক্ত write করতে পারবেন।

* আব যদির প্রথমে তখন কেউ read করতে পারবেন, তখন পোর্টের readers ও read করলে যেকোন write করতে পারবেন।

* at a time read / write একটাই হবে।

* Multiple readers → shared mode

* read - write problem টা semaphore
use করো।

1. writer Process :

at a time একটা writer কর নিবু।

Sub: _____

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

Date: / /

2. Reader Process:

* যদি কোনো writer আছে তাহলে readers access করতে পারবেনা, writers access না করবে।

* যদি writers না আছে তাহলে multiple readers allow করবে।

* multiple readers read-count দ্বা দ্বারা synchronization problem হবে, তাহলে at a time একটি reader কি read-count দ্বা দ্বারা পারবে।

* এটা maintain করতে "mutex" এর value র আয়ে depend করবে ইয়ে।

• wait (wtr) → এ হ্যান্ডেলে কোনো writer আছে বিলু থাকলে, access করতে পারবেন।

exist → ক্ষেত্রে অন্য读者 mutex এর উন্ট করবে।

* wtr accessible হলে 1st reader wait loop থেকে অব ইয়ে মাধ্যে এবং mutex accessible হলে মাধ্যে।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

* 1st reader wtr access করতে পারলে
wtr কর্তৃ দিয়ে।

* অন্য reader মধ্যে ~~wtr~~ call করে তাহলে
অন্য করতে হবে, এটা তা multiple
readers access করতে পারেনা। তারাত
wait(mutex) এ wait করবে।

* অন্যকে process কে ~~wtr~~ call করতে
অন্যে অবগুলো process block হয়ে যাবে,
তাই only first reader wait(wtr) call করবে,

* 1st reader wait(wtr) থেকে যের হয়ে যাব
তাহলে অন্য reader আর access করবেনা।
তখন ~~wtr~~ wait(mutex) করে অন্য readers
read করার chance পাবে।

* read-count করাতে আবার wait(mutex) use
করা হয়।

* যখন অন্য reader এর read করা হয়ে
তখন signal(wtr) call করা হয়।

* অন্য read এর read করা হয়ে না হওয়া পর্যন্ত
write করা যাবে।

Sub:

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

1st readers problem \rightarrow readers এর priority থাকি

* ফেম্বুর লিটের স্ট্যানভেশন রয়েছে।

2nd readers problem \rightarrow readers স্ট্যানভেশন রয়েছে,

writer স্ট্যানভেশন প্রোবলেম : writer infinite time পর্যন্ত উৎক্ষেপণ করবে।

writer pre-empt করতে পারেনা \rightarrow readers রয়েছে।
reader রয়েছে priority

2nd readers problem : current readers রয়েছে।
writer pre-empt করতে পারে (writer priority)
starvation \rightarrow readers.

Sat Sun Mon Tue Wed Thu Fri

Sub: _____

Date: / /

Dining - Philosophers Problem

Counting semaphore → resource count

binary semaphore এর array use.

dining - philosophers problem → counting semaphore
যাই সেম্পারেমফোন কিম্বা অ্যারে ব্যবহার করা যাবাবলৈ,

especially কোনো particular chopstick কে
identify করা যাবাবলৈ counting এ-সেম্পারেমফোন

1st এই হাতের ডান পাতের chopstick → available
then n বাবলৈ n বাবলৈ n বাবলৈ n বাবলৈ

counting semaphore এ যথ্যাত্মক chopstick
available check. কোনো কোনো chopstick available
আছে তা check কৰতে পাবে না, তাই binary
semaphore এর array use.

Sub: _____

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

deadlock হ্যাঁ:

যদি ৫ টা chopstick available . সবাই থাতের আন
পাশের টা নিয়েছে , সবাই wait করছে হাতের বাস
পাশের টা available হওয়ার ক্ষেত্রে , মা বেধনাই
হবেনা .

philosopher 1

wait (chopstick) \rightarrow সবাই জুড়ি এটা call করার
ক্ষমতা ,

Deadlock situation :

* at most 4 টা philosopher আসলে allow .

* যখন একজনের 2 টা chopstick available আছে
allow .

* mutual exclusion . critical section implement
wait & signal এর সাহিতে ,

* ক্ষেত্রে Number philosopher আছে তার তাও বাস
বিভিন্ন n n n n আন

page \rightarrow 243 (figure) দেখ