NBP

Date: 4-12-32 OS - Ch# 1 (lecture 1)

Introduction to Operating System.

Computer System:

- Integrated form of different components working together.
- Each component has it's ~~own~~ specific task to perform.
- Working together they give the required output to the user.

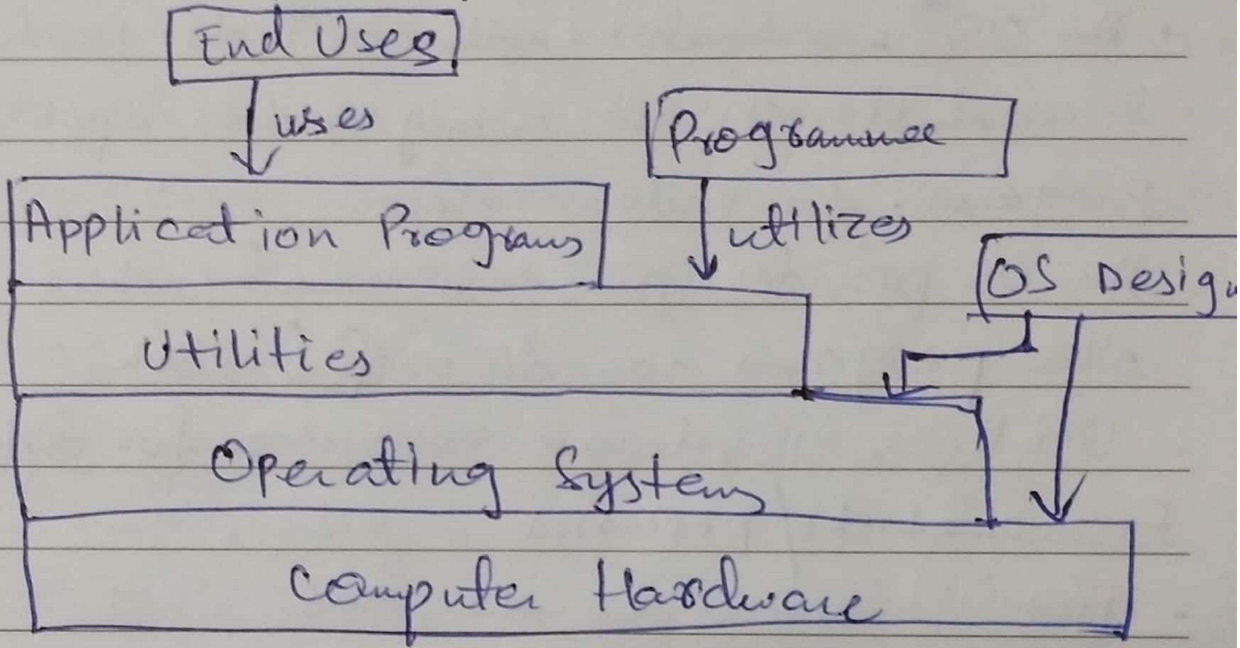
Computer System Structure:

1. ↳ Hardware: Basic computing resources
   e.g. CPU, memory, I/O devices --
2. ↳ Operating System: a system software. { controls the hardware & coordinates it among applications }
   e.g. Windows, Linux
3. ↳ Application Programs: They are used by our users, a way to use system hardware resources to solve any problem
   e.g. compilers, word processors etc.
4. ↳ Users: People, machine, other programs

14TH AUGUST HAPPY INDEPENDENCE DAY

Date:

Layers of Computer System.



What is an Operating System (OS)?

- A program that manages computer hardware.
- Provides basis for application programs.
- An intermediary between computer user & computer hardware.
- Some OS are designed to be convenient others to be efficient & some, combination of the two.

Date:

· What OS do?

· An OS ~~coordinates~~ ~~with~~ controls hardware & coordinates its use among various application programs for various users.

· An OS provides an environment where other programs can do useful work.

· Utilizes hardware resources for different ~~From the~~ tasks/processes.

· Provides services to ~~the~~ users

· Manages secondary memory & I/O devices

OS as a Resource manager.

From the computer's point of view, OS is much involved with hardware and a resource allocator.

CPU time, memory, file-storage space, I/O devices are all resources. OS manages these resources and provide them to solve a problem.

Date:

- There can be many requests for resources. An OS has to allocate them to the programs, so that the user can operate computer system efficiently.
- OS allocates resources to application programs across space & time.
  - → time sharing a resource (scheduling).
  - → space sharing a resource (allocation).
  - → CPU, memory, disk etc.
- There are limited resources and OS manages them efficiently.
  - ↳ improving utilization, minimizing overheads
  - ↳ improving throughput (tasks executed per unit time).
- OS is a control program.
  OS controls execution of programs to prevent errors & improper use of computer.

NBP

Date:

Services provided by the OS.

- Interacting with hardware is a complex or cumbersome task. Thus, an OS is an abstract machine that hides the complex hardware details underlying → system calls
- It provides common API's to users & system. An interface that makes it easy to utilize hardware resources.
- As a resource manager, it controls ~~provides~~, ~~control~~ access to shared resources. The resources shared b/w multiple process → inter-process communication. Resources are CPU processors, memory, disks, networks...
- Program development: Editor, IDE's → to make programs, debuggers
- Program execution: An OS handles how a program executes. loading it into memory, scheduling it's execution.

14TH AUGUST HAPPY INDEPENDENCE DAY

Date:

- Access to I/O devices: access to I/O resources in the hardware e.g. ~~Alt~~
OS provides device drivers (an interface) that programs can use to interact with hardware.
- Controlled access to files: grants access to a file to use it for data operations. It controls the restricted access such as read-only, read-write etc.
- System access: OS controls the restricted access of users to a system.
- Error detection & response: OS handles errors & fixes them. The error includes internal & external hardware errors.
  - → memory error → device failure.
  - → CPU → I/O devices etc.

Software errors.
  - → arithmetic overflow → access forbidden memory location
- Account: collect stats.
  - ↳ Task manager.

NBP

Date:

Basic element concerned with OS

① Processor: A processor controls entire computer system. It performs all type of computations.

② Main Memory: Primary memory, volatile memory e.g. RAM. It is used to store currently running programs & data.

③ Kernel: core/ nucleus of an OS. It provides, process & memory management, file system handling & device control. It is loaded into memory during system boot & remains resident to perform various functions.

④ I/O Modules: They facilitate communication b/w computer & peripheral devices. The OS interacts with I/O modules to manage data transfer & access. Secondary storage devices such as hard disk are utilized by OS ~~for long term data.~~

14TH AUGUST HAPPY INDEPENDENCE DAY

I said I would never fall
unless its yo, I fall into ♥

Date:

communication can be done with terminals (physical / virtual).

- System Bus: a communication pathway b/w CPU, memory an I/O devices. data transfer b/w these components.

One of the processor's function is to read or write data to or from memory respectively.
Two internal registers (registers are in processor) are use for this.

MAR: Memory Address Register, specified the address of memory from which data is to be read or write.

MBR: Memory Buffer register, it contains the data to write in memory or read an load into this register.

I/O AR: I/O Address Register → specifies any I/O device.

14TH AUGUST HAPPY INDEPENDENCE DAY

Po one

OS one jobs

Date:

I/OBR: I/O Buffer Register -> used to exchange data b/w I/O module & processor.

PC: Program counter -> holds address of next instruction to be executed.

IR: Instruction Register -> The instruction to be executed.

User-Visible Register.

They enable programmer to minimize main-memory refferences by optimizing register use. They can be reference by machine language

1. Data Registers
2. Address Registers
   - Index, Segment Pointer
   - Stack Pointer.

Index Register: add an index to a base value to get a new address.

Segment pointer: memory is divided into segments, then it is referenced by a segment or an offset.

Date:

Stack pointer: points to the top of the stack.

Control & Status Registers.

They are used by processor to control operation & execution of processor.

PC : address of next instruction.

IR : Instruction, most recently fetched

PSW : Program Status Word.

it contains control bits e.g. mode bit

it specifies privildged instructions.

Instruction Fetch & Execute.

These are set of instructions stored in memory.

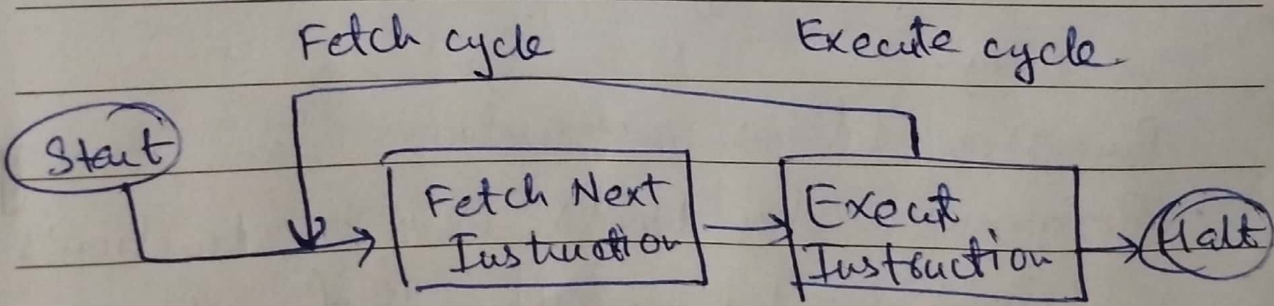
Processor fetches instructions from memory

Program counter holds next instruction's address & it is incremented after each ~~execution~~ fetch.

Date:

IR holds the instruction where it is decoded & then can be executed.
There are types of instruction

1. Processor - memory: Data transfer from processor to memory / vice versa.
2. Processor - I/O: Data transfer from to or peripheral device by transferring data b/w I/O modules & processor.
3. Data Processing: any ALU based operations on data.
4. Control: an instruction may specify that the sequence of execution can be altered.



Instruction - Execution Cycle.

Date: 5-12-23

Interrupts In OS (lecture 2)

- A mechanism by which normal sequencing of the processor is ~~an~~ interrupted
- Virtually, all computers provide this mechanism using other modules (I/O, memory)
- Interrupts are provided primarily as a way to improve processor utilization.
- An interrupt allows processor to execute another instruction while an I/O operation is in progress.
- An interrupt is an external event that stops the current execution of the process & can be continued where it left off after the interrupt is handled.

Classes of Interrupts:

① Program: generated due to some condition as a result of an instruction execution.
→ arithmetic overflow, I/O, out of bound

Date:

(2) Timer: generated by a timer within the processor. Allows OS to perform certain tasks regularly.
→ primitive ⇒ specified time slice
→ non-primitive ⇒ without any delay.

(3) I/O: generated by an I/O controller.
· signal normal completion of an operation
· signal multiple errors conditions.

(4) Hardware failure: power failure / memory parity error.

Types of Interrupts.

(1) Hardware Interrupts: A signal to CPU, triggered by external hardware devices or devices: to request attention.
e.g. keyboard input, mouse movements, or disk I/O.

Date:

2. Software Interrupts (Exceptions):
Generated via CPU in response to specific conditions. e.g. I/O or invalid memory access.

Types of Hardware Interrupts.

① Maskable Interrupts: can be delayed temporarily or delayed by the processor.
They handled on first come first served basis, sequentially.

② Non-maskable Interrupts: can't be delayed or ignored, demand an immediate attention.
handles priority-wise, nested.

NBP

Date:

Interrupt Handler.

A program that determines nature of the interrupt & what actions are needed to resolve it.

Whenever an interrupt signal is issued to the processor it pauses the execution of the user programe, take the PC value of next instruction to fetch & PSW (Program Status Word), which holds the current status of program & push it to the control stack. If there are any registers used by the program, there values will be pushed to stack & stack pointer is updated as may be the interrupt require those registers. Then the PC is updated to the point to the start of an interrupt service routine. Then the interrupt

Saad Butt

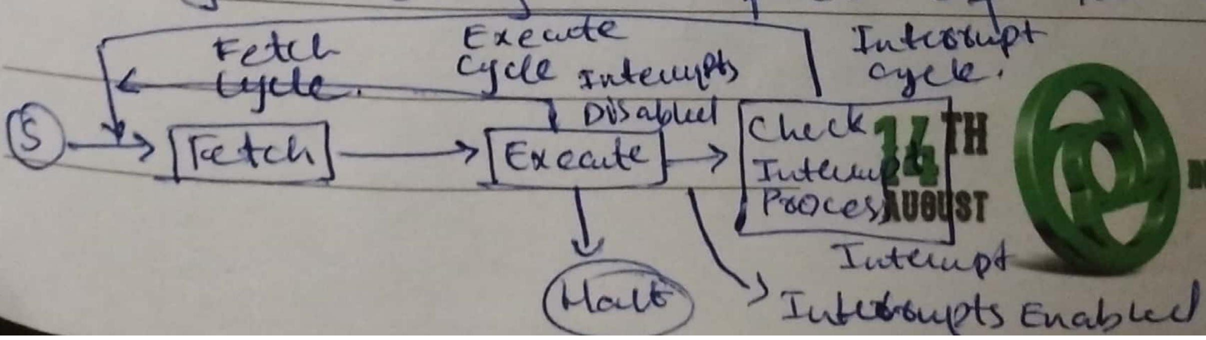
Date:

handler resolves the interrupt by determining type of interrupt & take actions to resolve it. When it is done, register values are retrieved back from stack and PSW or PC value as well or resumes the user program.

Interrupt Cycle:

On execution stage, processor checks for any interrupt signal. If there is none, it fetches the next instruction. If it finds one, it suspends the current execution of program. or executes an interrupt handler routine. ~~It is~~

Interrupt Handler Routine is generally a part of the OS.



Saad Butt

Date:

Multiple Interrupts.

One or more interrupts can occur while an interrupt is being processed.
One approach to handle this is that when an interrupt is being processed, disable the interrupts. Means, the processor ignores other interrupts, and they will be pending until processor reenables interrupts.

The drawback for this approach is that it doesn't handle priority. It can result in loss of data.

Second approach is to handle interrupts priority-wise. ~~A user~~

A user program is running.

An interrupt occurs. User program state info. pushed onto control stack & interrupt is being handled.

NBP

Date:

While an interrupt is being handled, another interrupt occurs with high priority. State information of first interrupt is pushed onto the stack & ISR of second interrupt is started.
In this way, first, second interrupt will be handled, then the first in then the user-program will be resumed.

Direct Memory Access (DMA):

The Memory Hierarchy:

How much is it → capacity.
How fast is it → access time
How expensive is it → cost

Fast access time, greater cost per bit.
Greater capacity, smaller cost per bit.
Greater capacity, slower access speed.

NBP

Date:

Going down the hierarchy.

- Decreasing cost per bit.
- Increasing capacity (storage space).
- Increasing access time. (more access time)
- Decreasing frequency of access of the memory by the processor.

locality of reference (address).

Registers are fastest, smallest and most expensive type of memory ~~consist~~. in processor.

Disk cache: A portion of main-memory used as a buffer to temporarily hold data.

Disk writes are clustered. Retrieval can be slow. Disk cache improves it.

Some data in disk may be referenced again, so for fast data access, it is placed in disk cache.

Date:

Cache Memory:

- Cache memory is invisible to OS, it interacts with other memory management hardware.
- It is a small, fast memory between the processor & main-memory. It is not visible.
- Main memory consists of 2^n addressable words (each Having a unique n-bit address).
- This memory consist of a number of fixed length K blocks.
  So, main memory has $2^n/k$ blocks.
- The cache consist of C slots or lines of K words.
- no. of C slots is lesser than main memory block's.
- If an instruction needs data & it is not in cache, then it will get it from main-memory & store the whole block to one of the lines in cache memory.

Date:

- principle of locality: ~~It says~~, the priority is high that if a data is accessed then its neighbour data will be accessed

Cache Design:

① Cache size: small caches have significantly ~~high~~ impact on performance

② Block size:

Block is a unit of data exchanged b/w cache & main-memory.

As block size increases, hit ratio increases. as of principle of locality.

But what if, the probability of newly fetched data becomes less than the probability of the data that has to be moved from main mem to cache. The hit ratio decrease.

③ Mapping function: When a new block is transferred from MM to cache, mapping function is used to determine which cache line, the block will occupy. Constraints are that which block will be replaced for new one.

Date:

(4) Replacement Algorithms: It identifies, which block is to be replaced, the one which in longer in the cache or has not been referenced.

LRU → Least-recently-used algorithm

(5) Write Policy: When a write instruction updates the data in the cache, it needs to be updated in ~~mem~~ MM as well

↳ can occur every time block is updated

↳ can occur only when block is replaced → less mem ops memory is obsolete

Direct Memory Access (DMA).

Techniques for I/O Operations.

(1) Programmed I/O:

- An instruction is executing or it requires I/O operation.
- Processor executes it by issuing the appropriate I/O module.
- The Programmed I/O, it fulfils the requested action & update the status register.

Date:

- But it doesn't alert the processor or interrupt the processor.
- Thus processor has to check periodically whether the requested I/O is completed or not. Thus processor has to wait for a long time.

?

② Interrupt-driven I/O:

- In this, the processor requests I/O module, cu do other work.
- The I/O module will interrupt the processor when the operation is done.
- Although, it is efficient than Programmed I/O but is still needs active intervention of processor to transfer data b/w memory & I/O module.

Date:

Both Programmed I/O & interrupt driven I/O have two inherent drawbacks
- I/O transfer rate is limited by speed.
- Processor has to manage I/O transfer some instructions must be executed for each I/O transfer.

(B) Direct Memory Access (DMA):
- The processor issues a command to DMA module the information.
  - → whether to read or write.
  - → address of I/O device involved.
  - → starting location in memory to read/write
  - → no. of words to read/write.
- After that the processor can do other work.
- DMA directly access memory & perform require operation

Date:

- When operation is done, DMA sends an interrupt signal to processor.
- For memory access, DMA needs system bus.
- Processor has to wait when it needs system bus as it is being utilized by DMA, so the process will pause for 1 bus cycle
- DMA is far more efficient than programmed or interrupt-driven I/O.

Multiprogramming:

- Processor has more than one program to execute.
- One processor doing multiple tasks or executing multiple programs at a time.
- The sequence ~~the~~ programs are executed depend on their relative priority & whether they are waiting for I/O.

14TH AUGUST HAPPY INDEPENDENCE DAY

Date: 6-12-23

~~UNIX Overview.~~

Lecture # 3: Evolution of OS.

Types of OS:

① Early Batch Systems: 1950.

② Simple Batch Systems: 1960

- No direct access of processor by user.
- A computer operator (a human) batches the jobs sequentially ~~&~~ on cards/tape to ~~a~~ batch on a Input device.
- A monitor is a software which uses the batch input.
- Monitor controls the sequence of events.
- Much of monitor ~~Re~~main in main memory → it is resident monitor.

Date:

- Monitor read the job one by one which it recieves through the I/P device.
- Then, that job is placed in user-program area & it is being controlled.
- When the job is completed, control is back to monitor & it reads next instruction in the same manner.
- When the batch is done, the result is send to an O/P device e.g printer to deliver it to user.
- The processor executes the instruction in main-memory.
- Processor will execute the instruction until it completes or get an error.
- In both cases, control is passed back to monitor to fetch next instruction from batch.

4TH AUGUST HAPPY INDEPENDENCE DAY

NBP

Date:

- Monitor performs a scheduling function so the jobs are in a queue.
- For each job, instructions are in primitive form of JCL (Job Control Language)
- JCL is a programming language used to provide instructions to monitor.
- Operator Driven Shop:
  An operator will batch all the jobs & give it to the input device then monitor & processor will do their work & produce an output to a device.

Q: How does the monitor know the nature of the job or which programs to execute?

Control Cards (Special card) that tells the monitor, which job to execute.

14TH AUGUST

HAPPY INDEPENDENCE DAY

Date:

There is difference b/w control cards and data or programs. $ sign distinguishes b/w them.

Q2: How does the monitor distinguish:
(a) job from job
(b) data from program?

Reading $FTN line, monitor load appropriate compiler. Then user program will be compiled into object code & stored in memory.
Reading $LOAD, monitor will invoke loader which loads the object code into memory.

JCL Instructions denoted by $

$JOB
$FTN
⋮ } Fortran Inst
$LOAD
$RUN
⋮ } Data
$END.

14TH AUGUST HAPPY INDEPENDENCE DAY

Date:

& transfer control to it. Here, the memory is shared for both, but one executes at a time.

Effects of JCL.

During execution of user program, any input instruction causes one line of data to be read.

input instruction invokes input routine.

input routine checks if a program has accidentally read JCL line.

If this happens an error occurs & control goes back to user.

Else, normal completion, monitor scans input lines & proceed with next JCL instruction.

Memory will be protected, because user program can alter memory & resident monitor lies in main-memory

The job will be aborted.

NBP

Date:

A timer is set with a range & if any job takes more time, it is stopped & control returns to the monitor.
Few machine level instructions are priviledged instructions & they'll be executed by monitor.

User mode: ~~what~~ in which user program are executed. Certain memory areas are protected, certain instructions may not be executed.
Kernel mode: where monitor executes. for priviledged instructions. can access protected memory areas.

. In batch OS, processor time alternates b/w execution of user program & monitor.

Date:

Resident Monitor.

- first OS –
- resides in main memory
- A job sequencer.
  - ↳ control is in monitor.
  - ↳ program is loaded, gains control.
  - ↳ job done, ~~th~~ control back to monitor.
  - ↳ automatically transfers control from one job to another, no idle time b/w programs.

Parts of Resident Monitor:

① Control Language Interpreter

↳ JCL interpreter, responsible for reading & carrying out instructions on cards (input cards).

② Loader: loads system programs & application programs into memory e.g. a compiler.

Date:

(b) Device Drivers: it knows special characteristics of system's I/0 devices

Spooling The use of secondary memory as buffer storage to reduce processing ~~storage~~ delays when transferring data b/w peripheral equipments & processors.

-> I/0 devices are slow.

-> I/0 & Processor could not overlap.

Solution

- Overlap I/0 of one job with computation of another job (processor) using double buffering, DMA etc.
- SPOOLing:

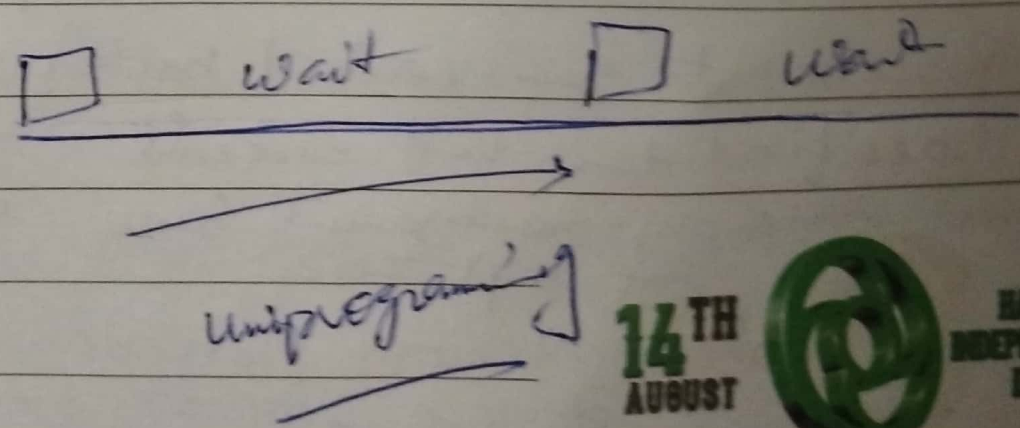
Simultaneous Peripheral Operation On Line

Date:

while executing one Job, OS reads next job in store it on disk.
A disk (Job Pool), it is a data structure, where OS selects which job will be running next.
It increases processor's utilization.

Uniprogramming: One program will run at a time using single processor.

- Processor must wait for I/O instruction to complete before proceeding
- more waiting time than execution time.
- latency increased.
- Poor utilization of processor with one program in memory.



Date:

Multiprogramming Batch Systems.

Multiprogramming:

When one job needs to wait for I/O, the processor can switch to other job, which we can say likely not waiting

The memory can be expanded n store more programs for switching.

It is multiprogramming / multitasking

Requirements for MP.

The hardware that supports multi-programing is interrupt-driven I/O ~~&~~ DMA.

I/O will be doing n ~~other~~ in the mean time. processor can do other job when, I/O is done, processor will be interrupted n interrupt-handler will handle it.

~~Use Using~~ Timer interrupts for Processor to gain control back when a Specified time limit exceeds.

Memory-management since several jobs are kept there.

Memory protection.

Software support from OS.

-> scheduling jobs.

-> resource contention

Date:

Why multiprogramming?

- Increased efficiency.
- It organizes job so processor always has a job to execute.
- more utilization of processor
- subset of jobs is kept in memory.
- Job scheduling manages jobs to execute one by one.

Requirements for Multiprogramming.

Done.

Time Sharing OS.

- Use multiprogramming to handle multiple jobs
- Processor time is shared among multiple users.
- Users simultaneous access through terminals
- OS interleaves among multiple users

NBP

Date:

- The goal is to maximize response time with multiprogramming
- In batch multiprogramming, goal is to maximize processor's use
  ↳ JCL provided with job
- In time sharing systems, commands entered by user on terminal.

Characteristics of Modern OS.

① Microkernel architecture:

Kernel has following functions.

→ address spaces

→ interprocess communication (IPC)..

→ basic scheduling.

communication b/w different processes.

Other OS services are provided by processes (servers).

14TH AUGUST HAPPY INDEPENDENCE DAY

Date:

② Multithreading.

A technique in which a process is divided into threads, that can run concurrently.

A Thread is a dispatchable unit of work.

Includes processor context, program counter & Stack pointer. In stack, the data used by that thread resides.

Thread executes ~~sequentially~~ parallely as it is interruptable (so that another thread can be executed).

③ Symmetric Multiprocessing : SMP

The OS & SMP schedules processes or threads across all processors

- multiple processors,
- share same main-memory & I/O facilities.

NBP

Date:

(4) Distributed OS:

illusion of single main memory &
secondary memory space &
distributed file system.

(5) Object-Oriented Design.

adding modularity to the kernel.
programmers can customize without
disrupting system integrity.

NBP

Date: 6 - 12 - 23

Lecture #4 OS Processes.

Processes.

→ A program in execution.
→ An instance of a program running on a computer.
→ The entity that can be assigned to & executed on a processor.

- Program code: it may be shared with other processes that are executing the same program.
- Set of data: data associated with that code.

These two are essential elements of a process.
While the program is executing, it can have no. of elements.
① Identifier: unique id to distinguish it from other processes.

14TH AUGUST HAPPY INDEPENDENCE DAY

Date:

→ 5 state model.

(1) State: Process is executing / running state

(3) Program Counter: address of next instruction in the program. PC

(4) Memory pointers: pointers to program code & set of data + any memory blocks shared with other processes.

(5) Context data: data present in registers in processor while process is executing

(6). I/O Status Information: i/O requests I/O devices concerned with the process, list of files use by process

(7) Accounting Information: may include processor time, & clock time used, time limits, account no.

This information resides in a data structure called Program Control Block (PCB), it is created & managed by OS

Date:

Process States:

Traces of a Process:

We can characterize the behavior of an process by listing the sequence of instructions that execute for that process. Such a listing is Trace of the process.

Dispatcher:

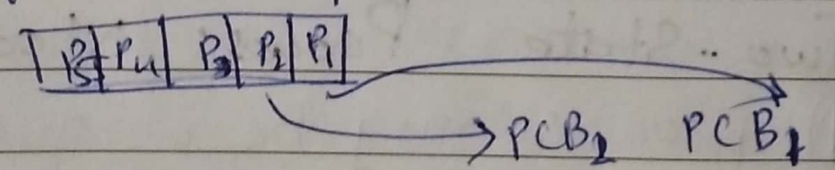
A small program that switches process from one to another.

Two-State Process Model.

- A process may be in one of two states. Running or Not Running.
- There are process cu OS knows it.
- One process is executing & time to time, it is interrupted & dispatcher of OS will select some other process

Date:

- This changes the first process state from Running to Not running in the other process, vice versa.
- To keep track of the process state OS keeps track of its current state or location in memory.
- ↳ Process Control Block.
- Processes not running are placed in a queue waiting to execute.
- There is a queue, in which each entry is a pointer to the PCB of a particular process



- Queue can be an implementation of a linked list / data blocks.
- If process is interrupted / timed-out / I/O operation occur, it will go into queue, otherwise exit the system.

Date:

Process Creation:

When a new process is added to those which are currently being managed, the OS builds the data structure which are used to manage an allocating space in memory.

Reasons

1. New batch job: a batch of job is done, new sequence of job called commands.
2. Interactive log-on: User logs-on to system from terminal.
3. Created by OS to provide a service: OS creates process on behalf of another application program.
4. Spawned by existing process: For modularity / to exploit parallelism, a process can request no. of processes.

Process spawning: OS creates a process at explicit request of another process.

Date:

Process Termination.

In batch jobs: Halt instruction will interrupt OS that the process is completed. The interactive appliction, a user maybe log off from terminal

There could be errors which cause termination.

Reasons for Process Termination:

① Normal completion: process executes an OS service call.

② Time limit exceeded: specified time limit reached.

③. Memory unavailable: process requires more memory then system allows.

④ Bounds violation: out of bond reference of memory.

⑤ Protection error: unprivilegedce, writing a read-only file, accessing a file which doesn't exist

NBP

Date:

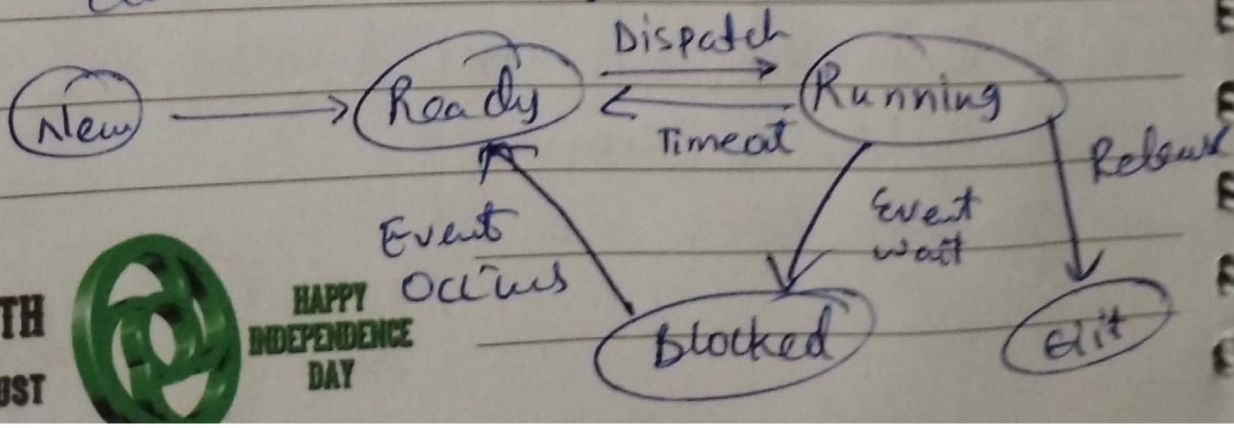
(6) Arithmetic error: I/O, storing number larger than accomodated.
(7) Time overrun: process has waited longer than specified time.
(8) I/O failure: inability to find a file.
(9) Invalid Instruction: process attempts to execute nonexistent instruction.
(10) Privileedged Instruction: process attempts to use instruction which is not allowed.
(11) Data misuse: wrong data type.
(12) Operator / OS Intervention: OS terminated the process.
(13) Parent termination: Parent process terminates so the child will.
(14) Parent request: parent causes child to terminate.

14TH HAPPY INDEPENDENCE

Date:

A five-state Model.

- processes ready to be executed
- FIFO queue in processor operates in round-robin.
- Each process in queue is given a certain amount of time, to execute & then returned to the queue, unless blocked.
- Not running processes are ready to execute.
- Some are waiting for I/O.
- Some are blocked.
- Dispatcher can't switch as it has to scan through the queue.
- Not Running states are broken down into Ready or Blocked.



Date:

(1) Running: Process currently executed.
(2) Ready: Process ready to execute.
(3) Blocked/Waiting: Waiting for an event to occur e.g I/O operation.
(4) New: Process just created but not added to the job pool or added into main memory but its PCB is created.
(5) Exit: Process completed. exited from job pool. Halted or aborted.

Suppose, process A is running and B is in blocked state due to an event. B's priority is higher than A. OS learns the event has occurred and B is in ready state. Then it'll interrupt process A and dispatch process B. Here, OS has preempted process B.

Date:

Two queues are maintained.
① Ready Queue -> for processes, ready to execute
② Blocked Queue -> for processes waiting for an event to occur.
n Blocked queues for hundreds or thousands of events.
There could be n Ready Queues if there are priorities in processes. One for each priority

Process Control Block (PCB)

- Each & every process in OS is representating by a PCB.
- A data structure in OS kernel containing information needed to manage a particular processor.

Date:

PCB, kernel level
Process identification
Processor State Information
Processor Control Information
User level
User Stack — Private user space for programs & data.

Shared address space.

Pointers: contains address of another processes present in ready queue.
Process state: information about process state eg. New, Ready, Running, Blocked Exit.
Program Counter: address of next instruction to be executed.
CPU Registers: Number of registers depending on Computer architecture eg accumulators, index registers, stack pointer, general purpose registers.

Date:

CPU Scheduling Information:
It includes
Process priority, Pointer to scheduling queues; Event: Identity of event the process is awaiting before it can be resumed. (in waiting state).

Memory management Information:
It includes value of base & limit registers.

Accounting Information:
It includes amount of processor & real time used, time limits, account numbers, job / process numbers etc.

I/O Status Information:
It includes list of I/O devices allocated to the process, list of open files etc.

Date:

PCB contains critical information about processes as it must be placed where it is protected from normal user. PCB is placed at ~~top~~ of kernel stack.

Interleaving Execution of Processes:

- Alternating between processes.
- Processes are executing priority-wise

Modes of Execution:

1. User mode: less priviledged mode for user programs.
2. System, control or kernel mode: more privilledged mode kernel of an OS.

Date: 12-12-23

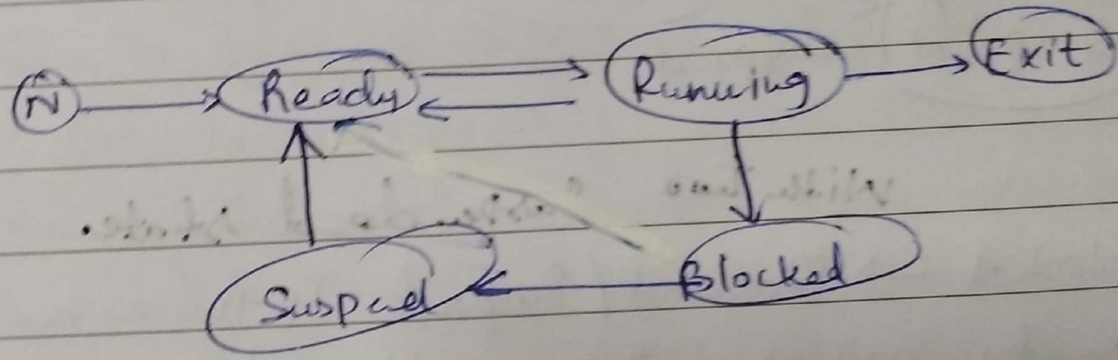
Suspended Processes:

- ~~As~~ each Process must be loaded fully into main memory
- Swapping, moving part or all of a process from main memory to disk.
- When process in main memory are not in Ready state, the OS swaps one of the blocked process out onto disk into a suspended queue. A queue for the processes, that have been temporarily kicked out from main-memory/suspended.
- Then a newly created process can come or one of the processes from suspended queues.
- Processor is faster than I/O, so processes must be waiting for I/O
- Processes in blocked state waiting for an event become suspended state and are moved to disk to free up memory.

Date:

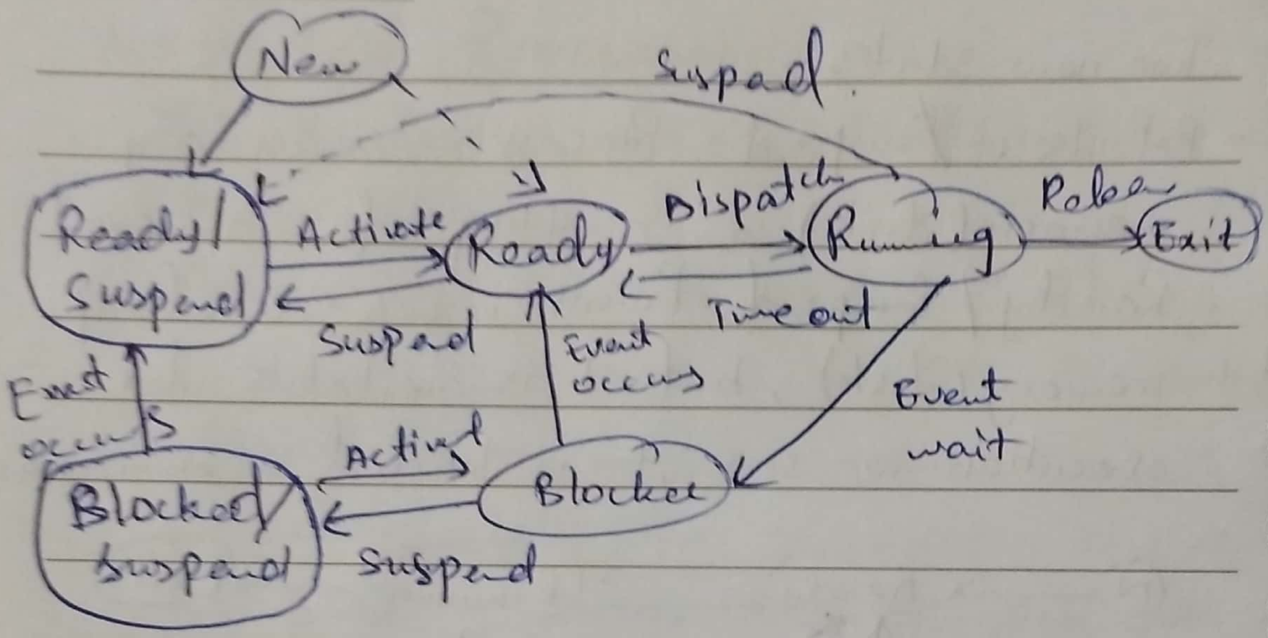
Two new states:

- Blocked/Suspend: Process in secondary memory (disk) and awaiting for an event.
- Ready/Suspend: Process is in secondary memory (disk), but it is available for execution as soon it is loaded in main-memory.



With one suspended state.

- Everything is same as five-state process model.
- Suspend state is added.
- From block, ~~it~~ process will move to suspend, to free up memory.



With two suspended state

- Blocked → Blocked/Suspend: If there are no ready processes, then one blocked process is swapped out to make room for another non-blocked process. or if any process requires more main-memory.
- Blocked/Suspend → Ready/Suspend: A process which is blocked and its event occurs then it is moved to Ready/Suspend State.

NBP

Date:

- Ready/Suspend → Ready: When there are no processes in Ready queue, one process will come to ready. or due to high priority
- Ready → Ready/Suspend:
  → ~~If a~~ ready Process is suspended, if it is only way to free up sufficient large block of memory
  → due to low priority
- New → Ready/Suspend or Ready: when new process is created, a PCB is made & allocate an address space to it. If OS does it in an early time, it would much MM, that's why, new process goto Ready/Suspend.
- Blocked/Suspend → Blocked: When a blocked/Suspend process has high priority than any of other process in ready or r/suspend, then it is reasonable to bring it back to Blocked.

14TH AUGUST HAPPY INDEPENDENCE DAY

NBP

Date:

- Running → Ready/Suspend: Due to preempting by OS for a high-priority process in blocked/suspend, that is now unblocked.

Reasons for Process Suspension:

1. Swapping: OS needs to make room of a memory for a process ready to execute.
2. Other OS reasons: Os may suspend a process, that is suspected of causing a problem.
3. Interactive user request: Suspended due to user's wish.
4. Timing: Process executing periodically is suspended for next time interval.
5. Parent process request: A parent process can terminate execution of a child process.

Date:

Os is an entity that manages the use of system resources by process.

What information does the OS need to control process & manage resources for them?

OS Control Structures:

- To manage processes and resources, OS needs to track record of each process and resource.
- OS constructs tables for each entity that is managing.
- Tables maintained by OS are:

① Memory table

- Keep track of main memory and secondary memory (virtual).

The memory table must include following

→ allocation of main memory to processes.

Date:

→ allocation of secondary memory to processes

→ Any protection attributes of both memories ~~for~~ access to shared memory regions

→ Inf. needed to manage virtual memory.

(2) I/O tables: used to manage I/O devices. It manages following

→ I/O device available or assigned to a process.

→ Status of I/O operation, if operation is in progress.

→ Location in main-memory, being used as source / destination of I/O transfer

(3) File tables: following information

→ Existance of files → Location in SM.

→ Current status → Other attributes

NBP

Date:

(ii) Process Table: to manage processes.

- All tables are linked to each other
- Memory, I/O & files are manage on behalf of processes
- There must be reference of these in process table.

OS manage & control a process by mainting the information following.

→ Where the process is located?

→ Attributes of process that are necessary to manage it (process id, state etc)

Process location

A process must include one or more programs to be executed. With this some data location for variables & constants. A stack for any procedure calls. Then some attributes of PCB (Process Control Block). Program, data, stack & attributes → process image

14TH AUGUST HAPPY INDEPENDENCE DAY

Date: 12-12-23

Lecture 5: Threads in Their Management.

Threads:

- The unit of dispatching | Unit of resource ownership is a process
- Light weight processes.
- A sequence of instructions, not a program, it cannot run on its own. as it is a part of a process.
- A process is divided in smaller units called threads.
- Smallest unit of execution,
- basic unit of CPU utilization.
- a process containing multiple threads, then they can execute in parallel.

Types of Threading.

① Single Threading

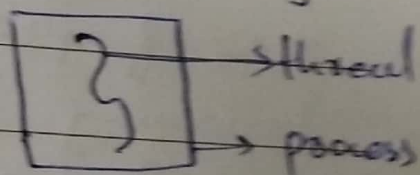
② Multiple Threading.

① Single Threading:

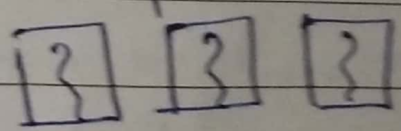
Date:

- Traditional approach.
- A single thread of execution per process
  Here, the concept of a thread is not recognized, if there is only one thread in a process, then what is the difference between process & thread.
- Example: MS-DOS supports single-threaded approach.

one thread per process



Some variants of UNIX supports multiple users per process but one process has one thread



multiple processes, one thread per process

Date:

7. Multithreading:

- Ability of OS to support multiple, concurrent execution within a single process.
- Windows, Solaris & Linux supports multithreading.

For single - threaded approach, in view of process management, there are following to mange,
① PCB ② User address space.
③ User Stack ④ kernel stact.

For multithreading.
only one ① PCB & ② User address space
But ③ user & ④ kernel stack are manged for every thread with a thread Control block for each thread.

Date:

Real-Life Examples of Multithreading:

① Web browsers:

-> One thread to retrieve data from server.

-> Another one to render that data.

Ex: Google, Firefox etc.

② Word Processors:

-> One thread to render text based material.

-> Another can manage oriatation of Page.

Ex: MS Office, Google Docs.

Benefits of Multithreading:

① Responsiveness: process responds much better while using multithreading.

② Speed up: process speeds up using multithreading, better experience to the user

Date:

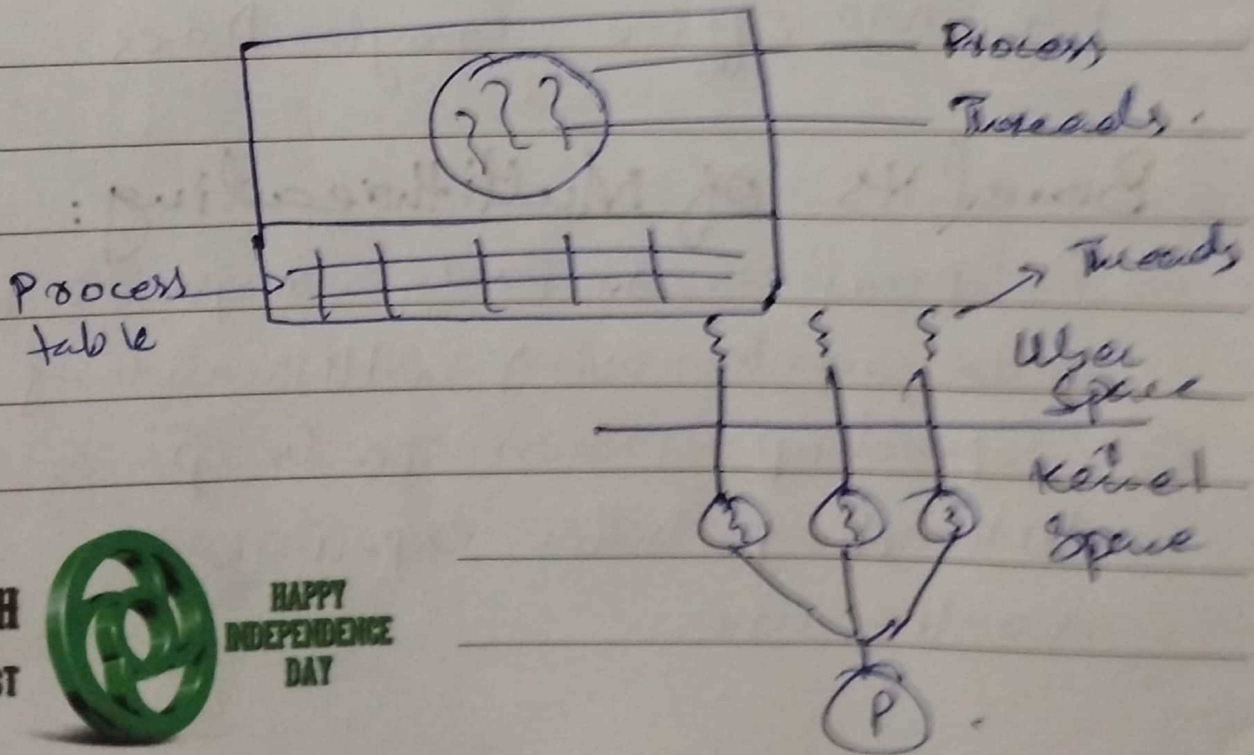
(3) Efficient Communication: Multiple threads of one process can communicate efficiently because of shared address space

(4) Utilization in multiprocessors: can take advantage of multiprocessor system

Implementation of Threads:

1- Kernel Level Threads: (KLTs).

- → Thread management is done by kernel.
- → Threads are implemented by kernel itself.
- → An API to kernel thread facility

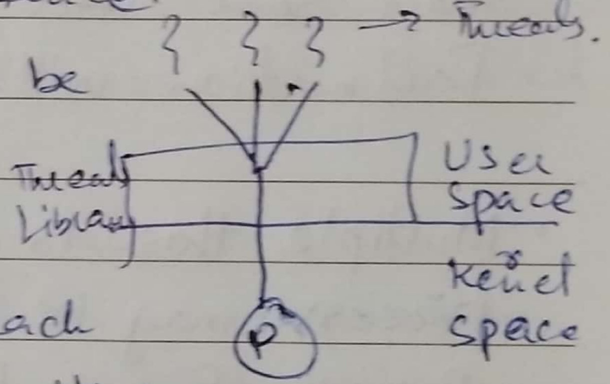


Date:

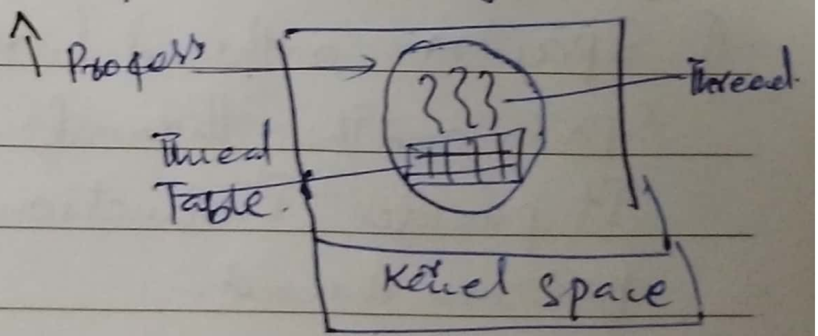
2. User-level Threads: ULTS.

→ All management is done by application, in user space. & kernel is not aware of thread's existence.

→ Threads library can be used to program multithreading.



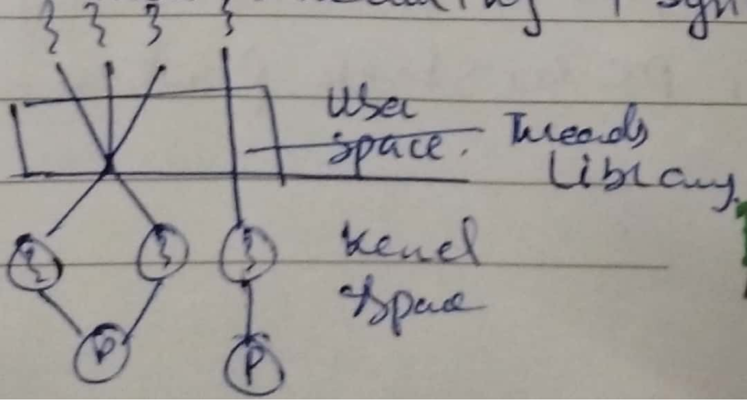
→ In user space, each process should have, its own private threads table. consist of PC, Stack pointer, Registers.



③ Combined Approaches ULT/KLT.

→ Thread creation in user space.

→ much scheduling & synchronization of thread



Date:

- If a process is suspended, all of its threads will also be. Since all threads share same ~~user~~ address space.
- Termination will also.

- Multiple threads within the same process may be allocated to separate process & executes simultaneously.

Thread States:

① Spawn: a thread within a process may spawn another thread.
It provides Instruction pointer & args for new thread.

② Block: When a thread needs to wait for an event, it'll be blocked, saving its registers, PC & stack pointer.

Date:

(3) Unblock: When the event due to which thread got blocked, occurs, it will be unblocked & placed in ready queue.
(4) Finish:- Thread completes, it register context & stacks are deallocated.

Remote Procedure Call Using Threads: RPC

RPC is software or distribute communication protocol in which one process (a client) requests another process (server) to communicate, synchronize & transfer information to & forth from source to destination.

14TH AUGUST HAPPY INDEPENDENCE DAY