OS

process execution → state changes
- new (1) → stored in HDD [queue]
- running (3)
- waiting (4) [queue] RAM
- ready (2) → RAM [queue]
- terminated (5)

(1) to (2) by long term scheduler (in sec, min) (infrequent)
(2) to (3) by short term scheduler (milliseconds) (freq.)
(3) to (4) by content switching
(4) to (2) by dispatcher (special short term scheduler)

task control block - information of process
(Process Control Block)
context switch → saves state of old process while reallocation
↳ does no useful work in mean time.

Process scheduler → schedule queue
- Job queue
- Ready queue
- device queue

waiting queue →
- I/O req. → I/O queue
- time slice expired
- fork a child
- wait for an interrupt

Process creation

UNIX →	creation	fork()	windows → create process()
	exit	exec()	
	delete	exit()	
	terminate	abort()	

exit() returns status data (child → parent) (via wait())
why abort()? →
- exceeds allocated sources
- task is no longer required
- parent is exiting. (cascading termination)

parent may wait to terminate (via wait()).
pid = wait (& status).
No parent waiting ⇒ zombie process, parent term. ⇒ orphan process without wait()

Google chrome browser (multiprocess)
- Browser (manages UI, disk, network I/O)
- Renderer (runs in sandbox restricting disk & network I/O)
- Plug-in

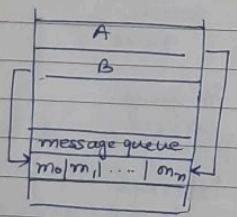
each tab represents seperate process

can be co-operative or independent

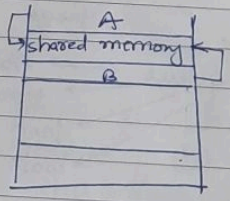
why co-op.?
- info sharing
- computation ~~speed~~
- modularity
- convenience

→ needs interprocess communication (IPC)

IPC (types)
- shared memory
- Message passing



message passing
(takes more time)
(manage physical & logical link)
[send (message)
recieve (message)]
(estb. a comm. link)
(exchange msg via send/recieve)
(through bus/network/internet)
(sync. issue (same link shared))
(link capacity)
(link is uni or bi directional)



synchronisation issue
(data incosistency)
(1 reads 1 writes at same time)
(~~no nee~~ fastest)
(hardware support needed)

Message passing
Physical link — either by — Shared memory
Hardware bus
Network

Logical link — Direct or indirect
Sync. or async.
Automatic or explicit buffering

Direct comm. send (P, message)
receive (Q, message)
automatically establishing link.
(associated with only 1 pair of comm. process)
(each pair → 1 link only)
(usually bi-dir., but maybe unidir

Indirect comm.
msg directed & received from mailboxes (ports)
(unique id
must be shared b/w 2process to comm.
(link estb. requisite)
(link associated with many process)
(each pair → many link)
(maybe uni or bi)

msg. passing — maybe → blocking (sync.) → blocking send - sender blocked till msg received
blocking recieve - reciever blocked till msg recieved
→ nonblocking (async.)

blocking (sync.)
- guarantee of succ. comm.
- both blocked till succ. verified. (dec. perf.)

Nonblocking send (sender sends msg & continue)

Non blocking receive (Receiver receives
: valid msg
: Null msg)

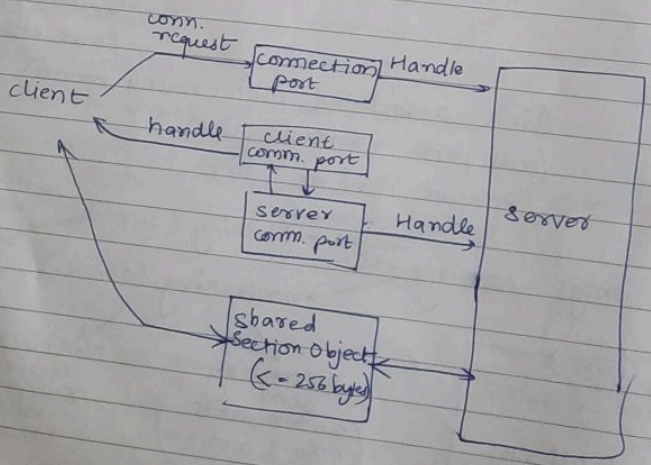
Rendezvous → both send & receive are blocking

Non blocking (async)
If error occurs, time consuming
more overhed

Buffering
Queue of msg attached to link
• zero capacity queue - msg must be read as soon as it is sent
• bounded capacity queue - finite length
• unbounded capacity queue - infinite length (sender never waits)

⇒ IPC system example

advanced local procedure call (LPC)
• processes on the same system
• uses ports



Comm. for process on diff. system.

- Sockets
- Remote procedure calls
- Pipes
- Remote Method Invocation (Java)

Socket

defn. → endpoint for comm, uses port & ip address to create link

eg 161.25.19.8 : 1625
(Ip address of host : port) = socket.

ports below 1024 are well known (used for standard services)

special ip address: 127.0.0.1 (loopback)
refer to system on which process is running

Types →
- Connection oriented (TCP)
- Connectionless (UDP)
- MulticastSocket class

Remote procedure call

- abstracts procedure calls b/w process on networked systems.
- uses port

Stubs - client side proxy for actual procedure on server

client-side stub → locates server
↳ marshalls the parameters

server-side stub → recives the msg
↳ unpacks marshalled parameter
↳ performs procedure on server

window stub code compile from MIDL
Microsoft Interface Definition Language

Data representation via XDL format
(External Data Representation)
for diff arch. Big-endian & Little endian

More failure in Remote comm. than Local
Message delivery → exactly once. (not almost once)
OS → provides rendezvous/matchmaker service to connect client & server.

Pipes
: acts as conduit for comm.
Ordinary - parent process creates a pipe to comm. w/ child process.
(cannot be accessed from outside)
Named - (can be accessed without parent-child rel^n.)

- Ordinary - comm. in producer-consumer style
  - producer writes (write-end of pipe)
    consumer reads (read-end of pipe)
  - unidirectional.
  - parent-child rel^n required.
  - anonymous pipes (as per windows)
- Named - More powerful
  - bidirectional
  - parent-chid rel^n not required
  - several process can use it.
  - supportive on both UNIX & windows

Threads (Light Weight Process)

content ~~context~~ switching time

to reduce size of process for less overhead and faster performance, by storing less parameter (as the size is small), less amt. of time to store process, as size of PCB is less, so less time to send data.

multithread server arch.

client → request → Server → create new thread for the request. → thread

Server → resumes listening for addn requests

- high responsiveness
- efficient resource sharing
- cheaper than process creation
- scalability – process can take advtg. of multiprocessor arch.

Multicore programming

challenges faced →
- Dividing activities
- load balancing
- Data splitting
- Data Dependency
- Testing & Debugging

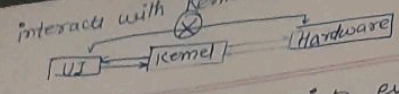
Parallelism: performing more than one task simultaneously

Concurrency: supports more than one task making progress (frequently switching).

→ Data parallelism: same op., diff. data.

→ Task Parallelism: unique op.

UI interacts with kernel which interact with Hardware.



multithread divides LWP into even smaller size making data transfer efficient (only need to send the thread containing required data)

Amdahl's law

S is serial portion
N is processing cores
1-S is parallel portion

$$\text{Speedup} \leq \frac{1}{S + \frac{(1-S)}{N}}$$

as N approaches ∞, speedup approaches $1/S$

Thread type → User thread
managed by user-level thread lib
- Posix Pthreads
- Windows threads
- Java threads

Kernel threads
Supported by Kernel
- Windows, solaris, Linux, Tru64 UNIX, Mac OS X

Multithreading Models - to Map User thread to Kernel thread.

Many to One | One to One | Many to Many

#

Eg:

Thread Library — Provide programmer with API for creating & managing threads.

implemented → whole library existing in user space
→ Kernel level Library supported by the OS

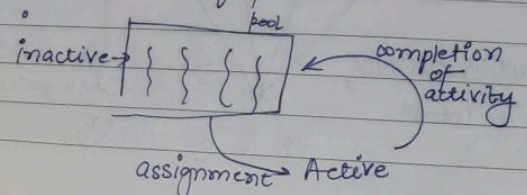
Pthreads
- provided at either level
- POSIX standard (IEEE 1003.1c) API
- specification, not a programming lang.
- common in UNIX OS. (Solaris, Linux, Mac OS X).

Implicit threading
- Thread pools (Windows)
- Open MP (LINUX)
- Grand Central Dispatch.

Windows use Microsoft Threading Building Blocks (TBB)

other - java.util.concurrent package

=> Thread Pools
No. of threads in a pool where they await work
- slighter faster (use an existing thread instead of creating one
- No. of threads in application bound to the size of pool



creation & termination time can be reduced.

1. Creation
2. Assignment
3. Termination