UNIT - 5

PROCESS AND RESOURCE MANAGEMENT

LECTURE NOTES

1) Process

* Process term is used in MULTICS s/m in the 1960. Process is an asynchronous activity. It is an active entity that requires a set of resources, including a processor, program counter, registers to perform its function. Multiple processes may be associated with one pgm.

* Process is a dynamic entity that executes a pgm on a particular set of data using resources allocated by the OS. A process can run to completion only when all requested h/w and s/w resources have been allocated to the process

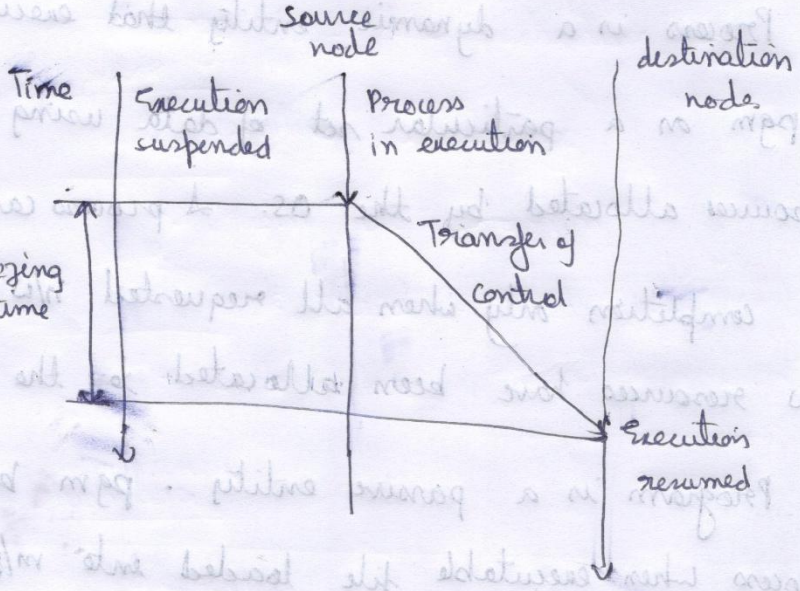
* Program is a passive entity. pgm becomes process when executable file loaded into m/y.

A process may be independent of other processes in the s/m.

2) PROCESS MIGRATION

* Process migration is the transfer of a sufficient amount of the state of a process from one computer to another for the process to execute on the target m/c.

* The following fig shows the flow of execution of a migration process



* Working of process migration:

1. Selecting a process to be migrated
2. Selecting the destination node
3. Suspending the process
4. Capturing the process state
5. Sending the state to the destination
6. Resuming the process
7. Forwarding future messages to the destination

* Process migration is of two types:

1. Preemptive process migration
2. Non-preemptive process migration.

* Features:

1. Transparency
2. Minimal interference
3. Minimal residual dependencies
4. Efficiency
5. Robustness

* Process migration mechanism

1. Freezing and restarting a process.
   a) Immediate blocking
   b) Postponed blocking
2. Transferring the address space and state
   a) There are three alternatives in transfer
      1) Total freeze
      2) Pre-transfer
      3) Transfer on reference
3. Message-forwarding mechanism
   1) Resending the message
   2) Origin site mechanism
   3) Link travel
   4) Link update

* Advantages.
   1) Balancing the load.

2) Moving the process closer to the resources it is using utilizes resources effectively and reduces n/w traffic

3) Being able to move a copy of a process on another node improves s/m reliability

4) A process dealing with sensitive data may be removed to a secure machine to improve security.

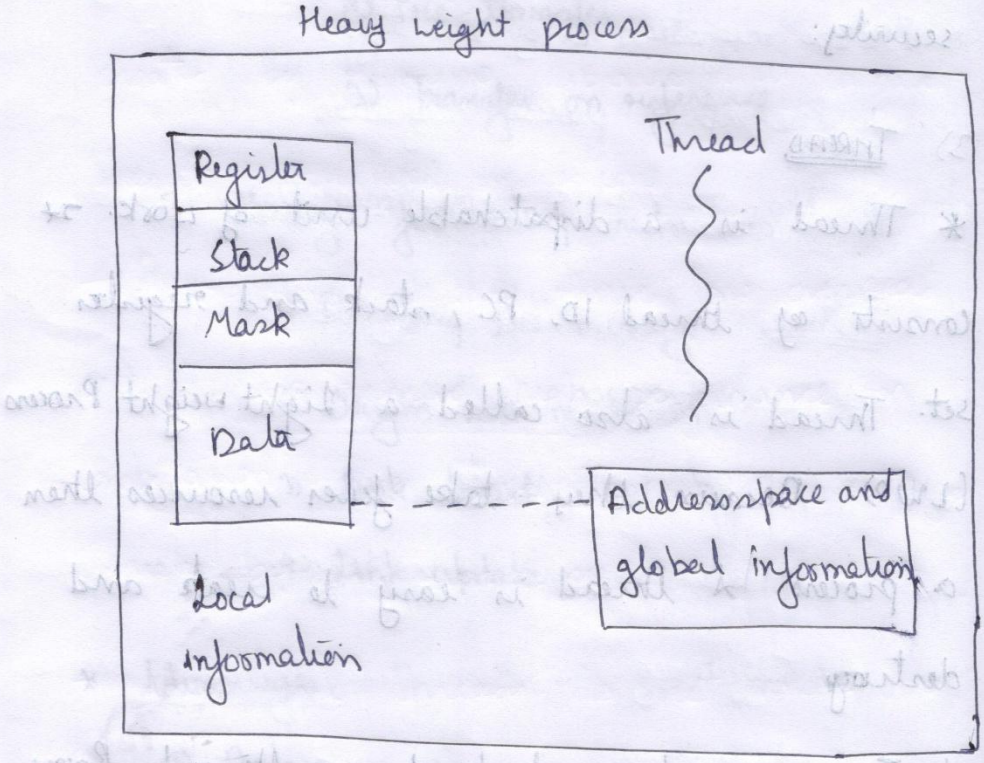
3) THREAD

* Thread is a dispatchable unit of work. It consists of thread ID, PC, stack and register set. Thread is also called a Light weight Process (LWP). Because they take fewer resources than a process. A thread is easy to create and destroy

* Every pgm has at least one thread. Pgms w/o multithreading executes sequentially. That

ss. after executing one instruction the next instruction in sequence is executed.

* Thread is a basic processing unit to which an OS allocates processor time and more than one thread can be executing code inside a process



* Different OS uses threads in different ways

1. Free m/y is managed by kernel thread in LINUX OS.

2. Solaries uses a kernel thread for interrupt handling.

* Thread Advantage:

1. Context switching time is minimized
2. Thread support for efficient communication
3. Resource sharing is possible using threads.
4. A thread provides concurrently within a process.
5. It is more economical to create and context switch thread.

4) THREAD MODELS

* Threads are of two types:

1. User level thread
2. kernel level thread

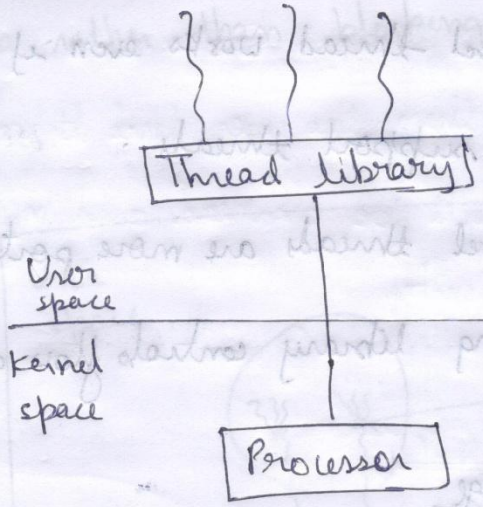
* All modern OS support threading model. Implementation of thread will change according to the OS.

* User level thread

→ User level thread uses user space for thread scheduling. These threads are transparent to the OS. User level threads are created by runtime libraries that cannot execute privileged instructions.

→ User-level threads have low overhead but it can achieve high performance in compilation. User-level threads are managed entirely by the run-time s/m.

→ User level threads do not invoke the kernel for scheduling decisions.



→ User level thread are also called many to one mapping thread because the OS maps all thread because the OS maps all threads in a multithreaded process to a single execution content. The OS considers as each multi-threaded processes as a single execution unit.

→ Advantages:

1) Kernel mode privilege does not require for thread switching.

2) These threads are fast to create and manage

3) User level thread works even if the OS does not support threads.

4) User level threads are more portable

5) Threading library controls flow of thread.

→ Disadvantage

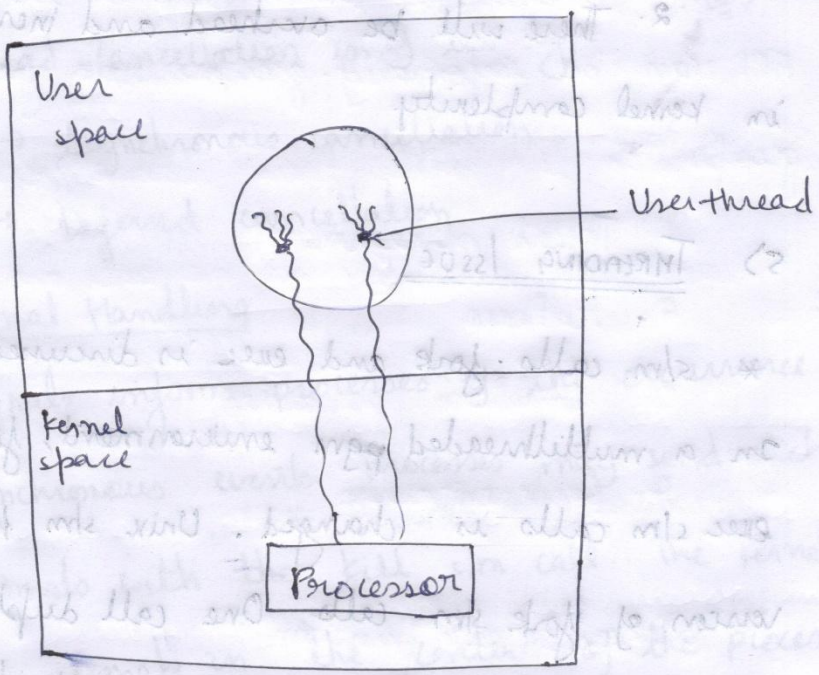
1. If thread blocks, the kernel may block the all threads.
2. Not suitable for multiprocessor s/m.
3. User level threads also do not support s/m wide scheduling priority.

* Kernel level thread

→ In kernel level thread, thread mangement is done by kernel. OS support the kernel level thread.

→ Since kernel managing threads, kernel can schedule another thread if a given thread

blocks rather than blocking the entire processes.



→ Threads are constructed and controlled by s/m calls. The s/m knows the state of each thread.

→ Advantages

1. Each thread can be heated seperately.
2. Kernel routines itself as multithreaded.

⇒

→ Disadvantage

1. Slower than the user level thread.

2. There will be overhead and increased in kernel complexity.

5) THREADING ISSUE

* s/m calls fork and exec is discussed here. In a multithreaded pgm environment, fork and exec s/m calls is changed. Unix s/m have two version of fork s/m calls. One call duplicates all threads and another that duplicates only the thread that invoke the fork s/m call. whether to use one or two version of fork s/m call totally depends upon the applns. Duplicating all threads is unnecessary, if exec is called immediately after fork s/m call.

(12) (SR) (13)

* Thread cancellation is a process of thread terminate before its completion of task.

* Thread cancellation is of two type
  → Asynchronous cancellation
  → Defered cancellation

* Signal Handling

→ Signals inform processes of the occurrence of asynchronous events. Processes may send each other signals with the kill sim call. The kernel handles signals in the context of the process that receive them so a process must run to handle signals. A signal may be received either asynchronously or synchronously.

Pattern of signals are as follows:

1) Generated signal is delivered to a process.
2) Once delivered, signal must be handled
3) A signal is generated by the occurrence of a particular event.

6) THREAD IMPLEMENTATION

* The subroutines which comprises the Pthreads API can be informally grouped into four major groups:

1. Thread management
2. Mutexes
3. Condition variables
4. Synchronization

* pthread_create function
* pthread_join function
* pthread_self function
* pthread_detach function
* pthread_exit function
* pthread_sigmask, pthread_kill
* Thread attribute

→ Steps for specify customized thread attributes

1. Create a pthread_attr_t object. The easiest way is simply to declare an automatic variable of this type.

2. Call pthread_attr_init, passing a pointer to this object. This initializes the attributes to their default values.

3. Modify the attribute object to contain the desired attribute values

4. Pass a pointer to the attribute object when calling pthread_create.

5. Call pthread_attr_destroy to release the attributes object. The pthread_attr_t variable itself is not deallocated; it may be reinitialized with pthread_attr_init

7) RESOURCE MANAGEMENT INTRODUCTION

* Distributed s/m contain a set of resources interconnected by a n/w. The processes are migrated to fulfill their ~~response~~ resource requirements and resource manager are to control the assignment of resources to processes.

* Resources are of two types
  i) Logical resource
  ii) physical resource

* Types of process scheduling techniques
  i) Task ~~man~~ assignment approach
  ii) Load - balancing approach
  iii) Load - sharing approach

* ~~Fro~~

(16) (17)

* Features:

→ Desirable features of a good scheduling algms are as follows.

1. No a priori knowledge about the processes.
2. Dynamic in nature
3. Quick decision making capability
4. Balanced s/m performance and scheduling overhead.
5. Stability
6. Fault tolerance
7. Fairness of service

8) TASK ASSIGNMENT APPROACH

1) Processes have been split into tasks.
2) Computation requirement of tasks and speed of processors are known
3) Cost of processing tasks on nodes are known

4. Communication cost b/w every pair of tasks are known.

5. Resource requirements and available resources on node are known.

6. Reassignment of task is not possible.

* Goals of task assignment algorithm:

1. Minimization of IPC costs
2. Quick turnaround time for the complete process.
3. A high degree of parallelism
4. Efficient utilization of s/m resources in general.

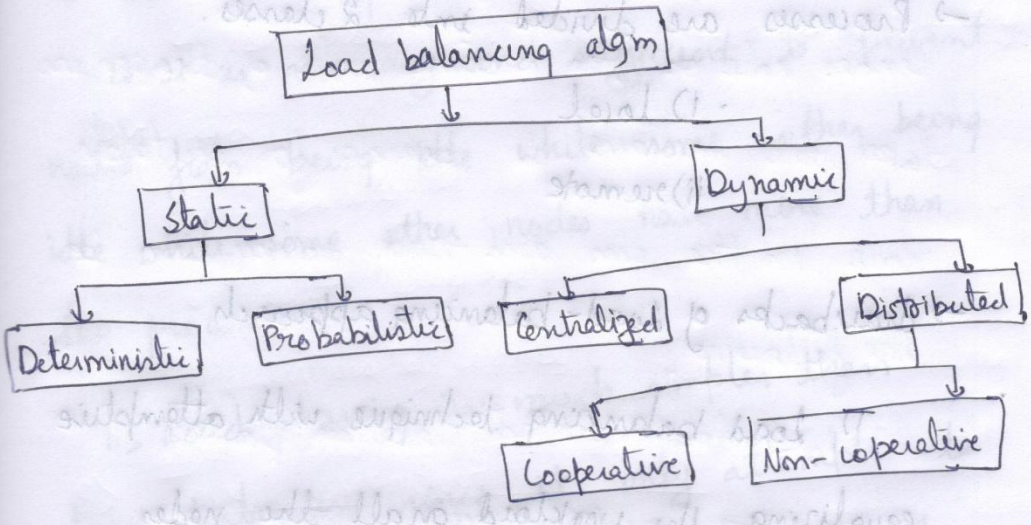
* Task assignment algms are of 3 types.

1) Graph theoretic deterministic algm
2) Centralized heuristics algm
3) Hierarchical algm

9) Load Balancing Approach

* Load balancing algm tries to balance the total s/m load by transparently transferring the workload from heavily loaded nodes to lightly loaded nodes.

* Taxonomy of load-Balancing algm



* Issues in designing Load balancing algm

→ Designing a good load balancing algm is difficult because of following reasons.

1. Load estimation policy
2. Process transfer policy
3. Location policy
4. Priority assignment policy
5. Migration limiting policy
6. State information exchange

→ Processes are divided into 2 classes:

i) local

ii) remote

→ drawbacks of load-balancing approach

1) Load balancing technique with attemptive equalizing the workload on all the nodes is not an appropriate object since big overhead is generated by gathering exact state information

2) Load balancing is not achievable since number of processes in a node is always fluctuating and temporal unbalance among the nodes exists every moment.

10) LOAD SHARING APPROACH

* Basic ideas for load-sharing approach is it is necessary and sufficient to prevent nodes from being idle while some other being idle while some other nodes have more than two processes.

ii) Load-sharing is much simpler than load-balancing since it only attempts to ensures that no node is idle when heavily node exists.

iii) Priority assignment policy and migration limiting policy are the same as that for the load-balancing algorithms.

Various issue in designing load sharing algorithm:

1. Load estimation polices
2. Process transfer polices
3. Location policies

→ Location policy decides whether the sender node or the receiver node of the process takes the initiave to search for suitable node in the s/m and this policy can be the following:

1. Sender-initiated location policy
   → Transfer policy
   → Selection policy
   → Location policy
   → Information policy
   → Stability

2. Receiver-initiated location policy :

→ Transfer policy
→ Selection policy
→ Location policy
→ Information policy
→ Stability

* Drawback

→ Polling initiated by receiver implies that it is difficult to find senders with new task.

→ Reason: s/m's try to schedule tasks as and when they arrive.

→ Effect: Receiver-initiated approach might result in preemptive transfers. Hence transfer costs are more.

→ Sender-initiated: Transfer costs are low as new jobs are transferred and so no need for transferring task states.

→ CPU scheduling algms are mostly round-robin, so a newly arrived task at an overload node quickly given a slice and this starts execution. Thus, very likely preemptive transfer of tasks will take place.