SNS COLLEGE OF TECHNOLOGY

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : **23CST202-OPERATING SYSTEMS**

II YEAR / IV SEMESTER

Unit 1-OVERVIEW AND PROCESS MANAGEMENT

Topic : Process Communication

Process Creation (Cont.)

There are also two possibilities in terms of address space of the new processes

- Child process is duplicate of the parent processes.
- Child processes has a program loaded into it.

- UNIX examples
- **fork**system call creates new process (duplicate of parent)
- **exec**system call used after a **fork**to replace the process' memory space with a new program.
- Wait() →If the parent has nothing to do while child runs (or till child completes) then parent can issue a wait system call to move itself off the ready queue until the termination of the child

Process Termination

When a processes terminates, all the resources of the process, including physical or virtual memory, open files and i/o buffers are de allocated by O.S

- When Process executes last statement and asks the operating system to delete it by implictly/explict call to the exit system call
  - Output data from child to parent (via **wait**).
  - Process' resources are deallocated by operating system.
- Parent may terminate execution of children processes (**abort**).
  - Child has exceeded allocated resources.
  - Task assigned to child is no longer required.
  - Parent is exiting.
  - Operating system does not allow child to continue if its parent terminates.
  - Cascading termination.
  - User could kill another process using the kill system call (kill generates a signal to abort a process but the process can ignore it (in most of the cases)
  - The wait system call returns the process identifier of a terminated child, so that the parent can tell which of it's possibly many children has terminated
  - If the parent terminates, however all the children are terminated by the O.S. Without a parent, Unix does not know to whom to report the activities of the child

Cooperating Processes

Independent process cannot affect or be affected by the execution of another process. (any process that does not share any data (temporary or persistent with any other processes is independent)

- ***Cooperating* process can affect or be affected by the execution of another process** (any process that share data with other processes is co –operating)
- Advantages of process cooperation
  - ☞Information sharing
  - ☞Computation speed-up (big program can be divided in many small program and they can run concurrently and can share data whenever required
  - ☞Modularity (logically breaking down a program in smaller ones make the processes/program more modular
  - ☞Convenience

Producer-Consumer ProblemTo be co-operative (share data) the process need to communicate →IPC (inter process communication)

A good example of co-operative process is →producer consumer problem

- Paradigm for cooperating processes, *producer* process produces information that is consumed by a *consumer* process.
- Example :
- producer = print program produces character, consumer = printer driver
- Producer = compiler produces assembly code, consumer = assembler
- Producer = assembler produces object code, consumer = loader
- The producer and consumer must be synchronized, so that the consumer does not try to consume an item that has not yet been produced. In this situation, consumer must wait until an item is produced
- Producer consumer need to share some memory (buffer) for information exchange these can be of two types
  - ☞*unbounded-buffer*places no practical limit on the size of the buffer.
  - ☞*bounded-buffer*assumes that there is a fixed buffer size.

Interprocess Communication (IPC)1) shared memory (as in producer consumer problem)2) Message system as discussed below

Mechanism for processes to communicate and to synchronize their actions

→definition of IPC.

- Message system –processes communicate with each other ***without resorting to shared variables***.
- IPC facility provides two operations:
  - ☞**send**(*message*) –message size fixed or variable
  - ☞**receive**(*message*)
- If *P*and *Q*wish to communicate, they need to:
  - ☞establish a *communicationlink*between them
  - ☞exchange messages via send/receive
- Implementation of communication link
  - ☞physical (e.g., shared memory, hardware bus (link))

Direct Communication

Processes must name each other explicitly (symmetric addressing):

- **send**(*P, message*) –send a message to process P
- **receive**(*Q, message*) –receive a message from process Q

- Asymmetric addressing : only sender need to name the receiver, the receiver is not required to name the sender { send (p, messages), receive (id, message), id will be automatically set the identity of the message sender }
- Properties of communication link
  - Links are established automatically.
  - A link is associated with exactly one pair of communicating processes.
  - Between each pair there exists exactly one link.
  - The link may be unidirectional, but is usually bi-directional.

Indirect Communication

Messages are directed and received from mailboxes (also referred to as ports).

- ☞Each mailbox has a unique id.
- ☞Processes can communicate only if they share a mailbox.
- ■Properties of communication link
- ☞Link established only if processes share a common mailbox
- ☞A link may be associated with many processes. (many members may be members of mailbox)
- ☞Each pair of processes may share several communication links. (some process may be sharing more than one mailbox)
- ☞Link may be unidirectional or bi-directional.

.

Indirect Communication

Operations

- create a new mailbox
- send and receive messages through mailbox
- destroy a mailbox

- Primitives are defined as:

send(*A, message*) –send a message to mailbox A

receive(*A, message*) –receive a message from mailbox A

Other IPC Issues

Process termination →if sender/receiver terminates there should be some way for the process (receiver/sender) at the other end to come to know about the termination

→(soln) Ack, multiple time outs

- Lost messages →(soln) Ack, time out (time out should be intelligently fixed otherwise duplicate may appear at the receiver and to solve this problem we will require duplicate filtering
- Scrambled message →use error checking codes like checksums, parity or CRC (cyclic redundancy check) and then tell the sender to resend if error is detected

THANK YOU