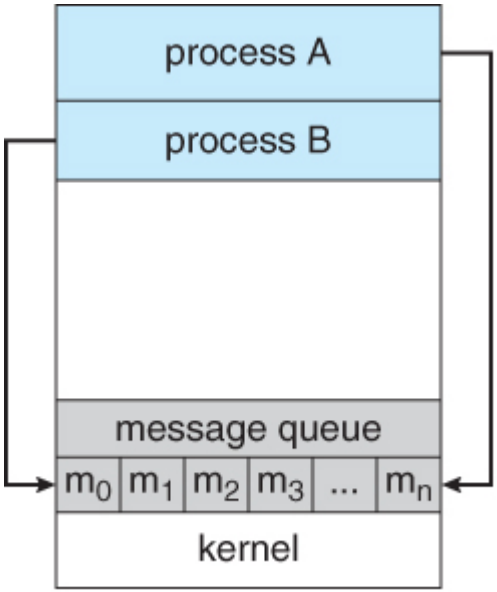
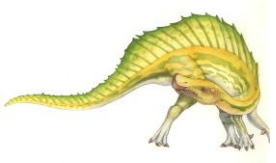
Interprocess Communication – Message Passing

- Second model of IPC (remember 1st was shared memory)
- 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*)





Implementation Questions

- How are links established?
- Can a link be associated with more than two processes?
- How many links can there be between every pair of communicating processes?
- What is the capacity of a link?
- Is the size of a message that the link can accommodate fixed or variable?
- Is a link unidirectional or bi-directional?

Shared Memory Example Producer

```
const int SHARED_MEM_SIZE = 4096;
const char *pSharedName = "CS460";

int main ()
{
 int segmentID;
 void *pSharedMemory;

 // Allocate shared memory segment
 segmentID = shm_open (pSharedName, O_CREAT | O_RDWR, 0666);

 // Specify the size
 ftruncate (segmentID, SHARED_MEM_SIZE);

 // Memory map the shared memory object
 pSharedMemory = mmap (0, SHARED_MEM_SIZE, PROT_WRITE, MAP_SHARED,
                       segmentID, 0);

  // Write message to the shared memory segment
 sprintf (pSharedMemory, "Love This Stuff!!!");

 return 0;
}
```

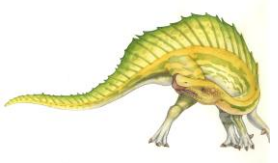
Shared Memory Example Consumer

```
const int SHARED_MEM_SIZE = 4096;
const char *pSharedName = "CS460";

int main ()
{
 int segmentID;
 void *pSharedMemory;

 // Allocate shared memory segment
 segmentID = shm_open (pSharedName, O_RDONLY, 0666);

 // Memory map the shared memory segment
 pSharedMemory = mmap (0, SHARED_MEM_SIZE, PROT_READ, MAP_SHARED,
                       segmentID, 0);

 // Read message from the shared memory segment
 printf ("%s", (char *) pSharedMemory);

 // Remove the shared memory segment
 shm_unlink (pSharedName);

 return 0;
}
```

Communications in Client-Server Systems

- Skipping Section 3.6.1 & 3.6.2

- Sockets
- Remote Procedure Calls
- Remote Method Invocation (Java)



Pipes

- Pipe - conduit allowing two processes to communicate

- Pipe issues to consider:
  - unidirectional vs bidirectional communication
  - if bidirectional, half duplex vs full duplex
  - parent-child relationship a must or not
  - communicate over network or not

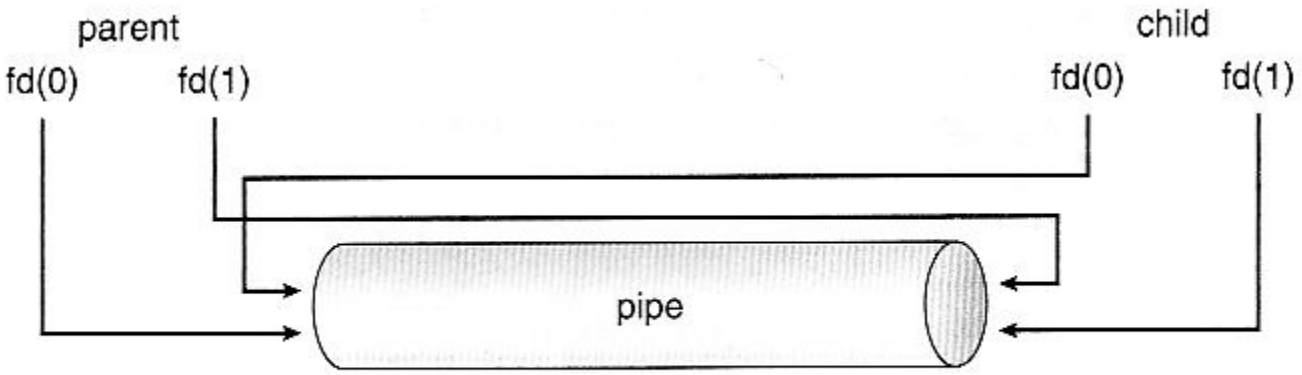


Figure 3.22 File descriptors for an ordinary pipe.

Ordinary Pipe

- Ordinary Pipe - communicates in standard producer-consumer fashion
  - write-end fd[1] file descriptor 1 is write
  - read-end fd[0] file descriptor 0 is read

```
int fd[2]; // pp. 143-144
pid_t pid = fork ();   // fork a child


if (-1 == pipe (fd))
{
  exit (1);
}
if (pid > 0) // parent write
{
  close (fd[0]);       // close read end
  write (fd[1], write_msg, lengthOfMessage);
  …
}
else    // child read
{
  close (fd[1]);       // close write end
  read (fd[0], read_msg, BUFFER_SIZE);
```

Main IPC Methods

Main IPC methods [edit]

Method	Short Description	
File	A record stored on disk that can be accessed by name by any process	Most operating systems
Signal	A system message sent from one process to another, not usually used to store information but instead give commands.	Most operating systems; some But other subsystems like the
Socket	A data stream sent over a network interface, either to a different process on the same computer or to another computer	Most operating systems
Message queue	An anonymous data stream similar to the pipe, but stores and retrieves information in packets.	Most operating systems
Pipe	A two-way data stream interfaced through standard input and output and is read character by character.	All POSIX systems, Windows
Named pipe	A pipe implemented through a file on the file system instead of standard input and output.	All POSIX systems, Windows
Semaphore	A simple structure that synchronizes threads or processes acting on shared resources.	All POSIX systems, Windows
Shared memory	Multiple processes given access to the same memory, allowing all to change it and read changes made by other processes.	All POSIX systems, Windows
Message passing (shared nothing)	Similar to the message queue.	Used in MPI paradigm, Java RI
Memory-mapped file	A file mapped to RAM and can be modified by changing memory addresses directly instead of outputting to a stream, shares same benefits as a standard file.	All POSIX systems, Windows

http://en.wikipedia.org/wiki/Inter-process_communication