

Interprocess Communication (IPC)

• Cooperating processes

- Often need to communicate



• Among threads (in same process)

- Shared address space (memory)
- Synchronization (more on that later)

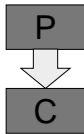
• For heavyweight processes

- Need OS support

IPC Applications

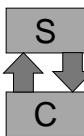
• Common architectures

- Producer/consumer
- Client/server



• Examples

- Calculation, compilation
- Database access, mail system



Message Passing

• Message

- Fixed size or variable



• Basic primitives

- Send message, receive message

• Need communication link

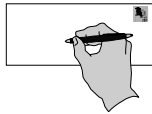
- Channel between processes
 - Typically managed by OS

Communication Link Issues

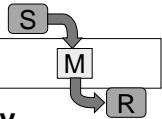
- **Link creation**
- **# processes per link**
- **# links per process pair**
- **Message size**
 - Fixed, variable
- **Buffering**
 - Capacity
 - Policy
 - Copy, reference
- **Directionality**
 - Unidirectional
 - Bidirectional

Link Partner Naming

- **Send: process ID?**
 - How to know recipient's ID?
 - Only one link between pair?
- **Receive**
 - Specified process to receive from
 - From any process
 - Sender ID delivered with message



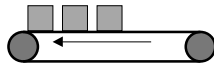
Indirect Communication



- **Messages via intermediary**
 - Mailbox, port, etc.
- **Sender to mailbox to receiver**
 - One writer, one reader
 - Multiple writers, one reader
 - Multiple writers, readers?
 - Reader block if no message to read?

Message Buffering

- **Zero capacity**
 - Sender, receiver must rendezvous
- **Bounded capacity**
 - Finite message queue
 - Sender must wait if full
- **Unbounded capacity**
 - No sender wait



Communication Modes

- **Asynchronous**
 - Sender, receiver access independent
 - Like mailing a letter
- **Synchronous**
 - Sender, receiver simultaneous access
 - Like a telephone call
 - Extension: remote procedure call

Exception Conditions

- **Process termination**
 - Sender or receiver
- **Lost message**
 - Network communication failure?
- **Error recovery**
 - Error-detecting codes, timeout, retransmission



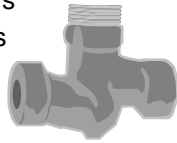
UNIX Pipes

• Interprocess communication

- Similar facilities in other OS's
- Pass more data than signals

• Related to file I/O

- More on file system later



• Normal pipes = parent/child

Creating a UNIX Pipe

• pipe system call

• Creates file descriptor pair

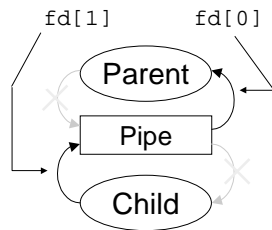
- Normally used with low-level I/O
 - E.g., open instead of file stream

• Unidirectional data

- Write second (1), read first (0)
 - Some UNIX variants are two-way

Parent/Child Pipe Setup

- Parent process executing
- Parent creates pipe
- Parent calls fork() to create child
- Parent closes fd[1]
- Child closes fd[0]
- Child writes to pipe, parent reads from pipe



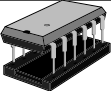


Using a Pipe as a File

- **Want high-level I/O**
 - Need `fstream`
 - `ifstream` or `ofstream`
 - Not `int` file descriptor
- **Convert to file stream**
 - Use (non-standard) constructor overload
 - E.g., `ifstream (int fd);`

IPC in Windows NT

- **Differs from UNIX**
 - No parent/child process relationship
 - No (unnamed) pipes
- **IPC mechanisms**
 - Named pipes (through file system)
 - Remote procedure calls (RPC)
 - Merges IPC and network messages

OS Resource Allocation

- **Processor** 
 - CPU time
- **I/O (e.g., disk)** 
 - Disk accesses (n/second)
- **Memory**
 - Allocated to process

CPU Scheduling

Allocation of CPU (time) resource

- Fair sharing
- Maximum throughput
- Minimum response time
- Predictable performance
- Minimize overhead

CPU Scheduling

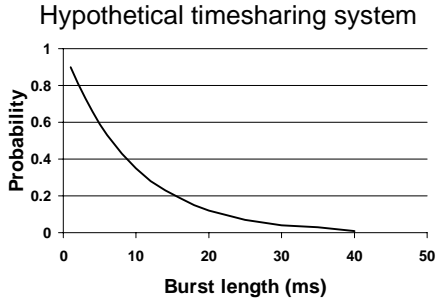
H. M. Deitel, *Operating Systems*

- Balance (all) resource use
- Optimize response vs utilization
- Avoid infinite delays
- Enforce priorities
- Degrade gracefully with load

Process Characteristics

- CPU-bound
 - Little I/O waiting
 - Would consume CPU if permitted
- I/O-bound
 - Short bursts of CPU use
 - Often interrupted by I/O waits

CPU-Burst Distribution



Schedule vs Dispatch

- **Dispatch**
 - Low-level function (context switch)
- **Schedule**
 - Short-term (CPU) scheduler
 - Choose next process to run
 - Long-term (job) scheduler
 - Assign priorities to processes
 - Select eligible group of processes?

Scheduling Types

- **Non-preemptive**
 - Process runs till blocked or completed
 - Not interrupted by OS
- **Preemptive**
 - Running process interrupted
 - May be forced to wait (ready state)
