



# UNIX IPC

Box Leangsuksun  
XCR@LaTech



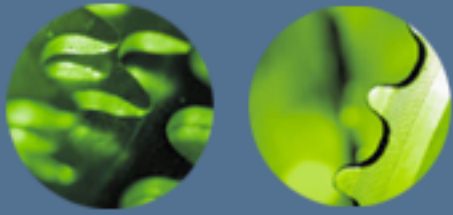
# UNIX IPC

- Read/Write to a file
- Pipe (at command line, pipe, mknod)
- Message Queue (mailbox concept)
- Semaphore (special
- Shared Memory



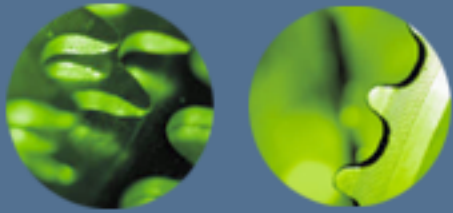
## Pipe

- unnamed FIFO structure for simple process communication
- `Cmd1 | cmd2`
- Normal steps
  - Create a pipe
  - Duplicate a pipe I/O streams to child/parent's ones and close the unused one
- See sample code



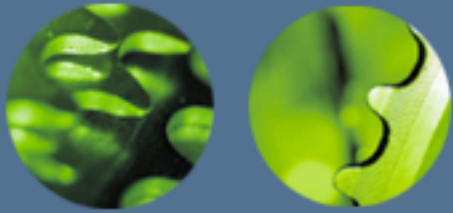
## Named FIFO (pipe)

- similar to Unix pipe but have a name.  
Create
  - “/etc/mknod name p”
  - mknod (2) , e.g.
    - `mknod (“/tmp/mypipe”, S_IFIFO, 0)`
- simply open a file for read (server) and write (client)
  - `fd = open (“/tmp/mypipe”, 0)`
- Use unlink to clean up

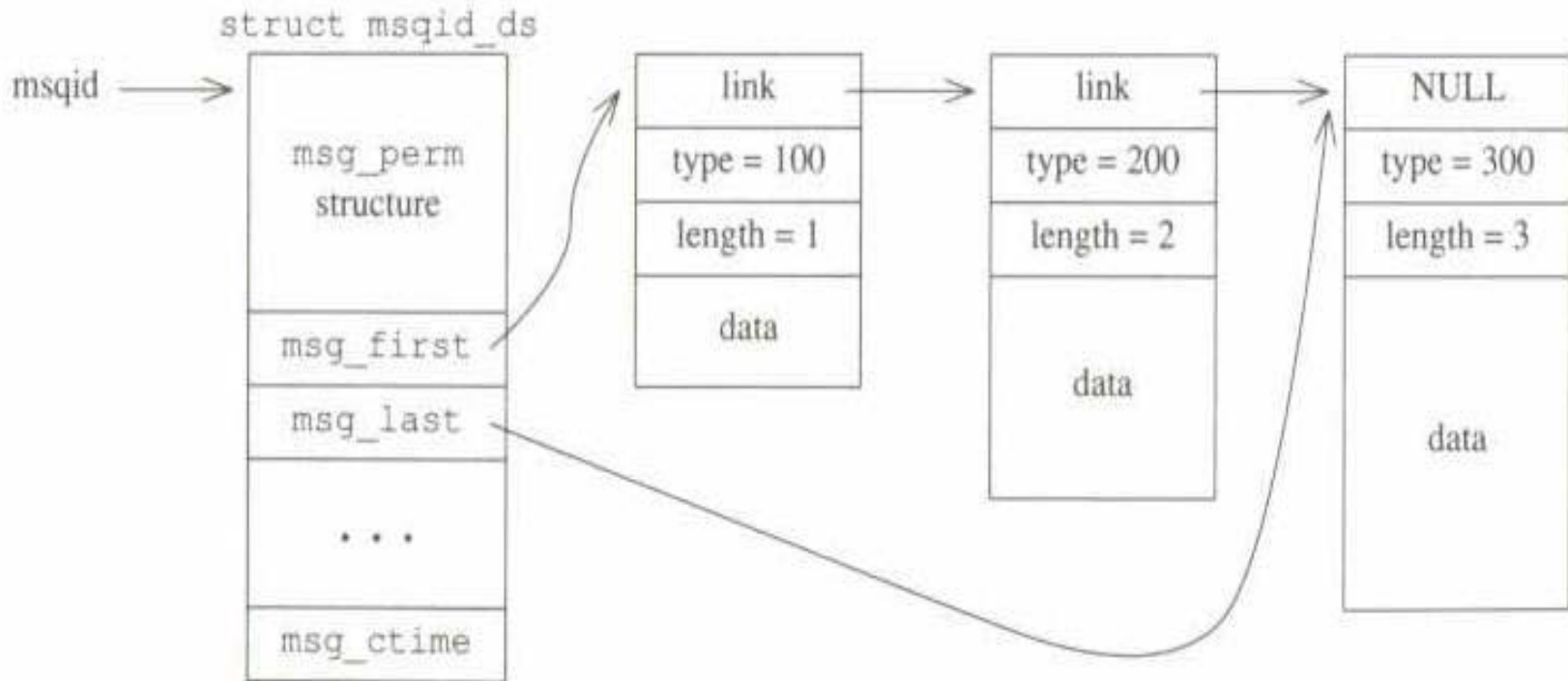


# Message Queue

- Similar to mailbox concept in OS book
- In kernel structure
  - Msgqid (key identifies which queue)
  - Each message
    - Type
    - Length
    - Data
  - Create a msg queue with `msgget()`,
  - Send/receive with `msgsnd()`, `msgrcv()`
  - Remove a message queue `msgctl()`



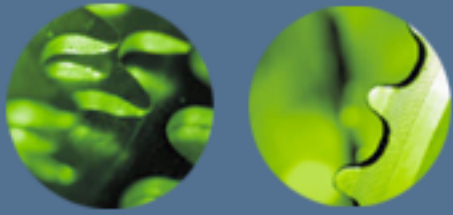
# Msg queue structure





# Semaphore

- Synchronization primitive (integer value)
- In kernel and use to synchronized the access to shared memory
- Set of semaphore operation guaranteed for atomic
- Should not be used to exchange large amount of data
- `Semget()`, `semop()`



# Shared memory

- Not in the kernel
- Used with semaphore
- Speed up the access to shared info
- Operations
  - Create : `shmget()`,
  - Attach to shared memory: `shmat()`
  - Do something (normal C/C++ operations)
  - Detach : `shmdt()`
  - Remove: `shmctl()`



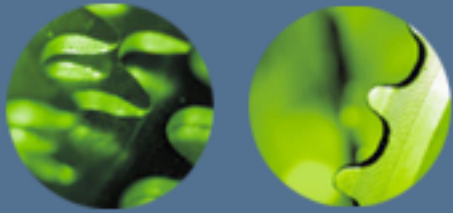
# Sample code

- **Server**

- Create a sh mem
- Attach to it
- Create 2 semaphores
  - Client will start first (1)
  - Server initialized to 0
- Wait to get a request (by reading a filename from a shared mem)
- Open a file and write content into a shared mem

- **Client**

- Get a share
- Attach to it
- Get a semaphore
- Write a filename
- Inform a server to start via server semaphore
- Wait to get a turn to read a file content



# Sample man page

- NAME
- `msgget` - get a message queue identifier
- SYNOPSIS
- `# include <sys/types.h>`
- `# include <sys/ipc.h>`
- `# include <sys/msg.h>`
- `int msgget ( key_t key, int msgflg )`