Processes 4: Misc Calls

1. Creation and Termination
2. Collecting Children
3. Exec-Family Calls
4. **Misc Calls**

Other Process-Related System Calls

We have seen the primary syscalls for process management:

- process creation (`fork()`)
- process termination (`exit()`, `_exit()`)
- process collection (`wait()`, `waitpid()`)
- program execution in a process (the "exec" library calls)

Additional process-related syscalls are often required in *multi-process, concurrent* programs, to provide:

- **interprocess communication (IPC)**
- process **synchronization**

Interprocess Communication (IPC)

Since processes have separate address spaces, when they need to exchange information, **interprocess communication (IPC)** mechanisms must be used.

Linux/UNIX provides a number of IPC mechanisms:

- **pipes**
- **FIFOs (named pipes)**
- **sockets** (IP vs. UNIX/local; stream vs. datagram)
- **shared memory** (POSIX vs. System V
- **memory mapping** (mapped file vs. anonymous mapping)
- **message queues** (POSIX vs. System V)
- **signals**
- temporary files

Process Synchronization

Multi-process, concurrent programs will typically require the use of **synchronization** mechanisms.

Synchronization may be required to ensure operations among the processes occur only in desired orders, or to prevent multiple processes from simultaneously accessing **shared resources**.

Linux/UNIX provides several *synchronization mechanisms* that can be used among processes:

- pipes/FIFOs
- signals
- semaphores
- file locks
- wait for child termination