

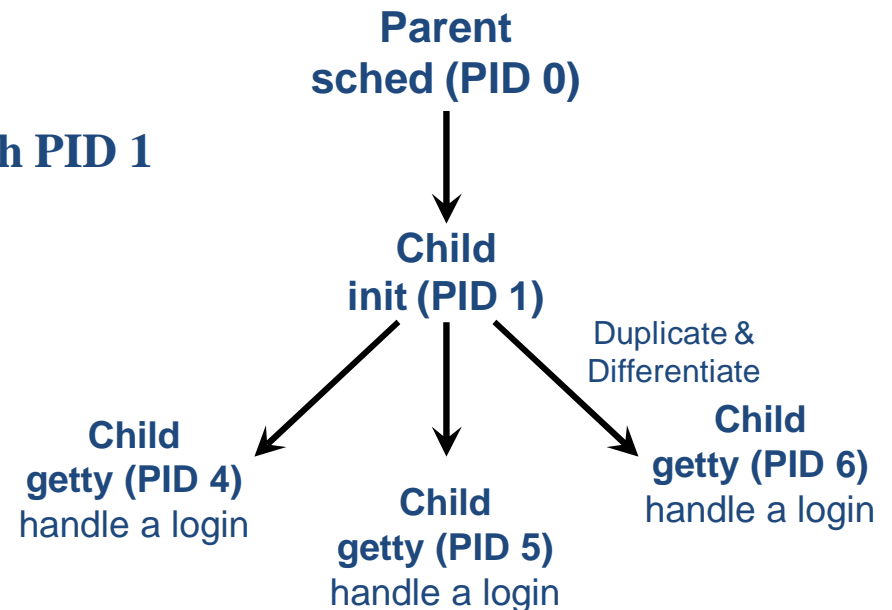
Unix Processes

Prepared by :
Swapan Purkait
Director
Nettech Private Limited
swapan@nettech.in
+ 91 93315 90003



Unix Processes

- In **UNIX** every process belongs to a process hierarchy
 - When **UNIX** is first started, there's only one visible process in the system
 - » **SCHED** process with **PID 0**
 - **SCHED** creates **INIT** process with **PID 1**
 - » **SCHED & INIT** are the ancestors of all other processes



Creating Processes

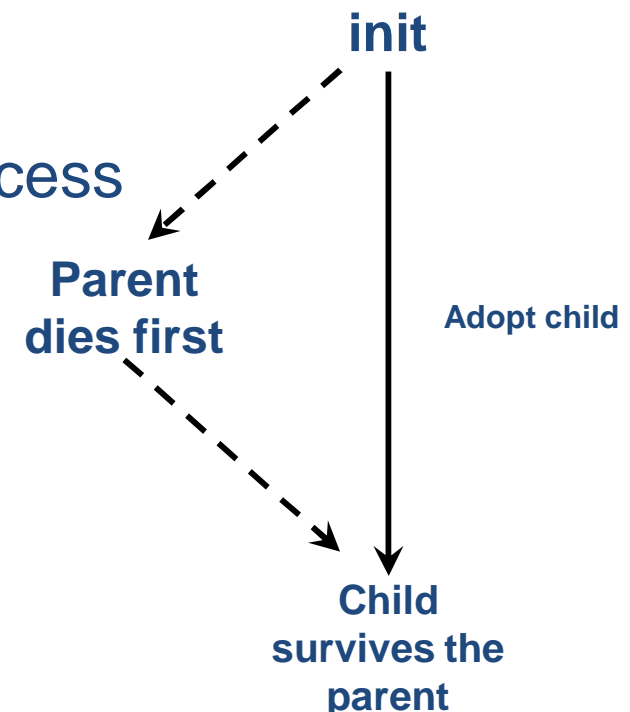
- **The only way to create a new process in UNIX is to duplicate an existing process**
- **fork causes a process to duplicate**
 - `int fork();`
 - **Returns**
 - PID of the child to the parent process (if successful)
 - 0 to the child process (if successful)
 - -1 to the parent process (if not successful)
- **The only difference between the child and parent processes is their PIDs and their PPIDs (parent PID)**

Process Identifiers

- **A process can use the**
 - **getpid function to get its own process identifier**
 - **int getpid();**
 - **getppid function to get its parent process's identifier**
 - **int getppid();**

- **What if a parent dies before its child?**
 - **The child is automatically adopted by the original “INIT” process**

- Parent process of the child
- becomes the “INIT” process



Terminating Processes

- **exit terminates a process by**
 - closing all of its file descriptors, and
 - deallocating its code, data & stack
- `int exit (int status);`
- **The status of the last terminated process can be accessed using the echo command**
 - `echo $status`

Zombie Processes

- **When a child process terminates, it**
 - sends its parent a SIGCHLD signal, and
 - waits for its termination code *status* to be accepted
- **A process that is waiting for its parent to accept its return code is called a *zombie* process**
 - It doesn't have any code, data, stack
 - However, it continues to inhabit the system's fixed-size process table

Zombie Processes

- **If a process's parent is alive, but never executes a wait(), the child process's return code will never be accepted**
 - **So the child process will remain a zombie process**

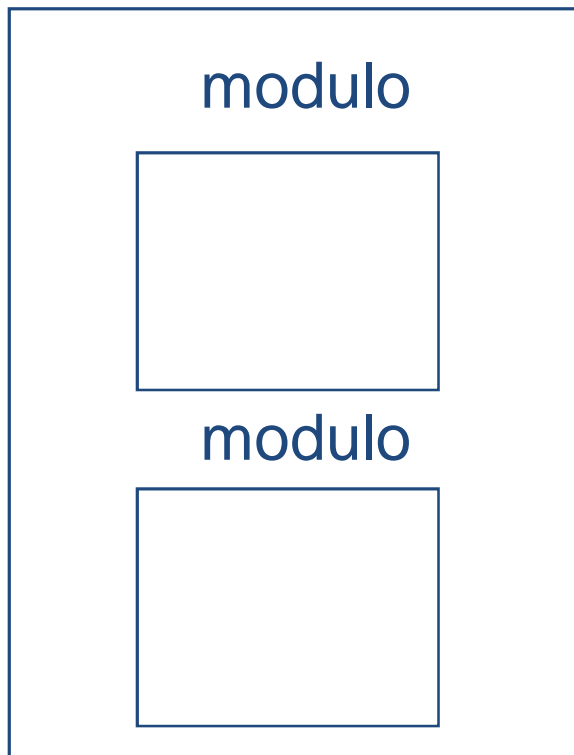
Waiting For a Child

- **wait causes a process to suspend until one of its children terminates**
 - `int wait (int* status);`
 - **Returns the PID of the child that terminated**
 - **Places a status code into *status***
 - If the rightmost byte of the *status* is zero, the leftmost byte contains the value returned by child's `exit ()` / `return ()`
 - Otherwise, the rightmost seven bits are equal to the number of the signal that caused the child to terminate, and the remaining bit of the rightmost byte is set to 1 if the child produced a core dump

IPC in UNIX

(InterProcess Communcation)

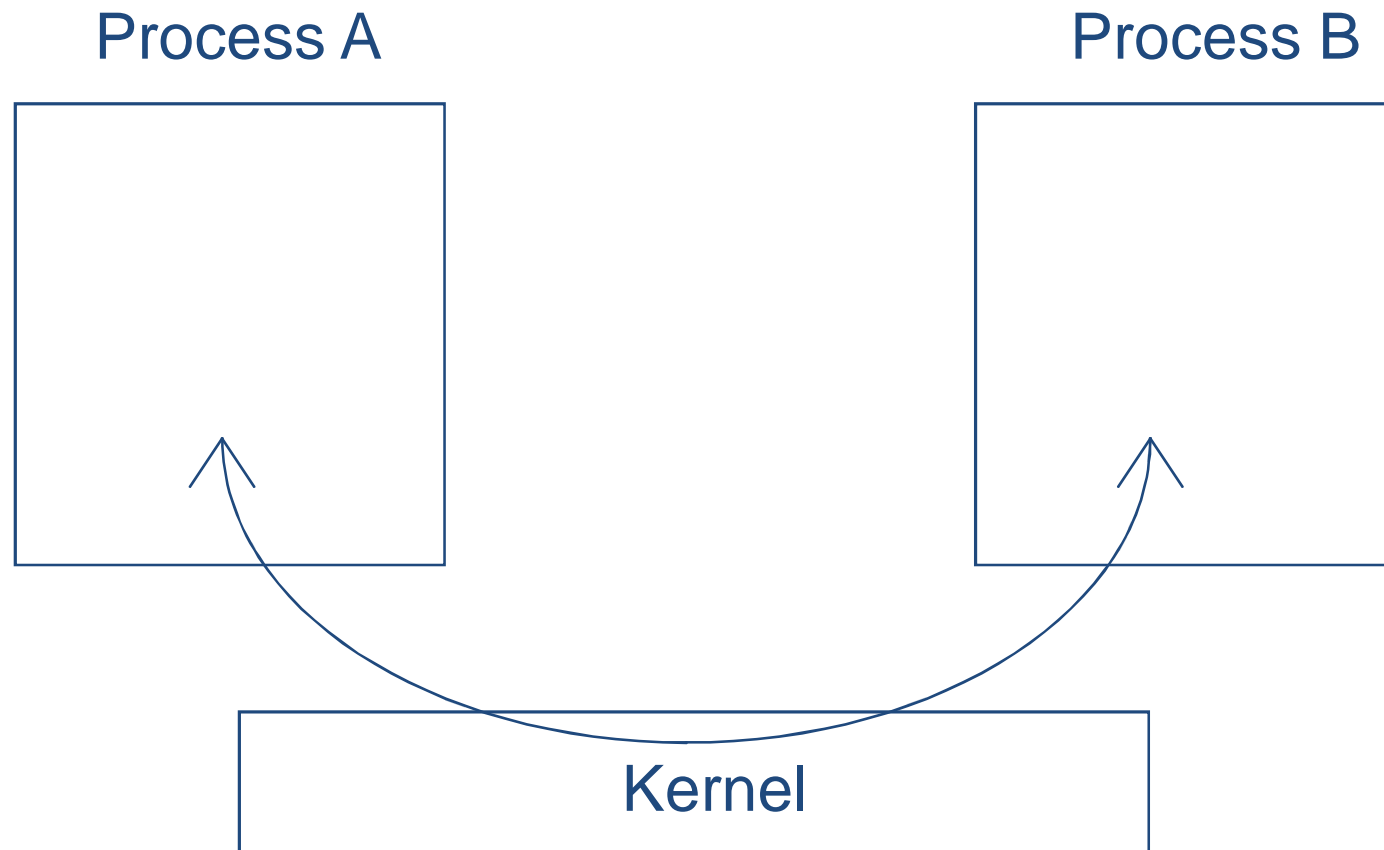
One Process



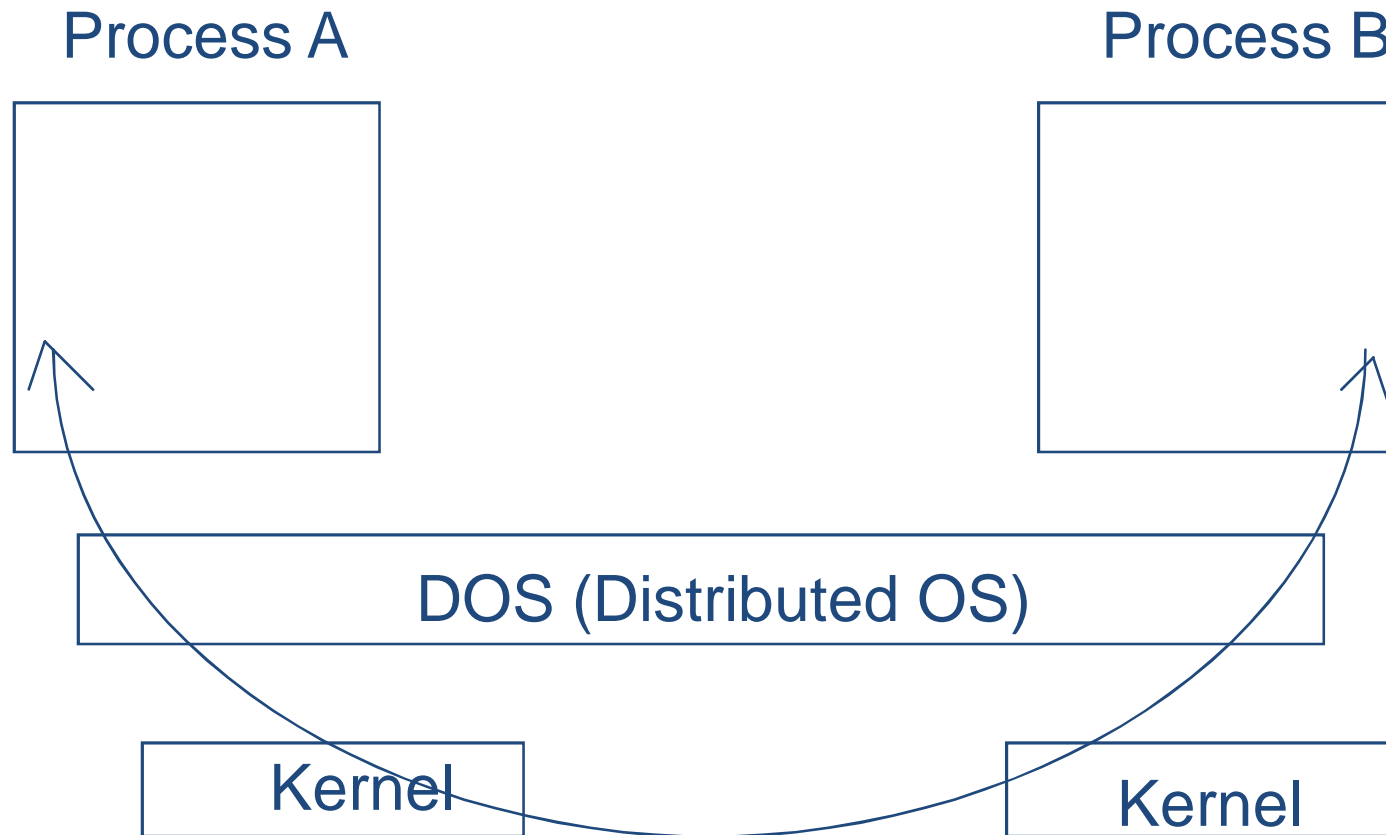
Communication of modules in a process

- global variables
- function calls
 - parameters
 - result

IPC On a Single Computer



IPC on Two Computers



Several Different Methods of IPC (in UNIX)

- Signals (Software Interrupt)
- Pipes and FIFOs
- Message Queues
- Semaphores
- Shared Memory
- Sockets
- RPC (Remote Procedure Calls)
- Mutex and Conditional Variables

Signals (on Unix)

- Software interrupts
- A notification to a process that an event has occurred
 - usually the process doesn't know ahead of time exactly when a signal will occur
- Signals can be sent by:
 - a process to another process (or to itself)
 - the kernel to a process

Signals Generation

- Every signal has a name specified in <signal.h>
Some examples of how signals are generated.
 - (1) The kill command can be used to send signals.
 - (2) Certain terminal characters generate signals
 - an interrupt character (<cntrl>-c or Delete) generates a signal called SIGINT
 - a quit character (<cntrl>-backslash)
 - terminates a process
 - generates a signal called SIGQUIT to process and generates a core image of the process
(a core file image can be used with a debugger for analysis)

Signals Generation (cont)

(3) Certain hardware conditions generate signals

- floating point arithmetic errors generate a signal called SIGFPE

(4) Certain software conditions can generate signals

- SIGURG signal is generated when some urgent data arrives on a socket

What can a process do with a signal?

1. A process can provided a function that is called whenever a specific type of signal occurs. This function, called a signal handler, can do whatever the process wants to handle the condition. (“catch” the signal)
2. A process can choose to ignore the signal. All signals, except SIGKILL, can be ignored.
 - SIGKILL is special, it guarantees the system administrator a way of terminating any process

What can a process do with a signal?

3. A process can allow the default to happen. Normally, a process is terminated on receipt of a signal with certain signals generating a core image of the process in the current working directory.

How to specify the signal handler?

- Use signal system call
 - i.e. `#include <signal.h>`
 - Signal is a function that returns a pointer to a function that returns an integer.
 - The function argument specifies the address of the function that doesn't return anything.
 - There are two special cases that provide special values for the function argument
 - `SIG_DFL` to specify the signal is to be handled in the default way
 - `SIG_IGN` to specify that the signal is to be ignored.
- If the signal handler returns, the process that received the signal continues where it was interrupted. (same as `ctxsw()`)

```
sigex.C
#include <signal.h>
main()
{
    int catchint();

    signal (SIGINT, catchint);
    printf("sleep call #1\n");
    sleep(1);
    printf("sleep call #2\n");
    sleep(1);
    printf("sleep call #3\n");
    sleep(1);
    printf ("Exiting \n");
    exit(0);
}

/* trivial function to handle SIGINT */
void catchint (int signo)
{
    printf("\nCATCHINT: signo=%d\n", signo);
    printf("CATCHINT: returning\n\n");
}
}
```

No user signals generated (i.e. <cntrl> - c)
> sigex

sleep call #1
sleep call #2
sleep call #3
Exiting

>

User generates a signal.

> sigex

sleep call #1

(user presses <cntrl> - c)

CATCHINT: signo = 2

CATCHINT: returning

sleep call #2

sleep call #3

Exiting

>

Thank you



Connect with us at Facebook

Visit www.facebook.com/nettech.in