TATA CONSULTANCY SERVICES

Experience certainty.

## Tata Code of Conduct

We, in our dealings, are self-regulated by a Code of Conduct as enshrined in the Tata Code of Conduct. We request your support in helping us adhere to the Code in letter and spirit. We request that any violation or potential violation of the Code by any person be promptly brought to the notice of the Local Ethics Counselor or the Principal Ethics Counselor or the CEO of TCS.  All communication received in this regard will be treated and kept as confidential.

TATA CONSULTANCY SERVICES
Experience certainty.

TATA CONSULTANCY SERVICES

# vILP – CPP – Operating System

## Managing Process Part 2

Version 1.1

**July 2015**

(ILP Guwahati Centre)

**Table of Content**

## *1.1 Parent and Child process*

A process can initiate a subprocess, which is a called a child process, the initiating process is referred to as its parent.

The child processes, in turn create other child processes forming a tree of processes ( which can be displayed using ps command with –forest option) A child process initially is a replica of the parent process, later the level of sharing of its resources and execution procedure may vary

>> Resource sharing

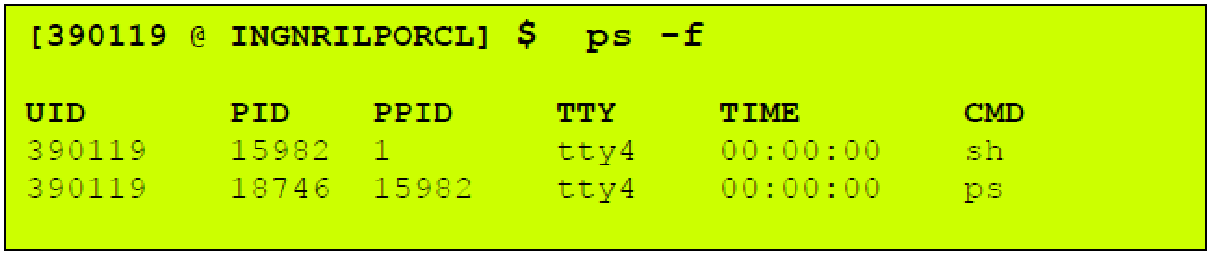
- all
- some
- nothing

>> Execution procedure

- Parent and child execute concurrently
- Parent waits for child to terminate
- Parent terminates prior to child process, which continues to execute

The init process inherits processes whose parents have terminated (Unix)

The command ps -f which displays the PID(process id) and PPID(parent process id) may be used to identify process and its parent process.



```
[390119 @ INGNRILPORCL] $  ps -f

UID         PID     PPID      TTY       TIME         CMD
390119      15982   1         tty4      00:00:00     sh
390119      18746   15982     tty4      00:00:00     ps
```

## *1.2 Orphan Process*

When a child process is killed, parent process gets the notification via a signal. Parent then, can continue other task. However if the parent process is killed before, its child is called an orphan process. In such cases init process(parent of all process) becomes(adopts) the new parent of the child.

## *1.3 Zombie Process*

When a process finished its execution and exit status not received by the parent ( or parent did not read the exit status till now), the process state becomes zombie.

The process is dead (not to be scheduled for further execution) but cannot be completely removed from process table, until it has been determined that exit status is no longer needed.

## *1.4 Daemon Process*

Some programs are not designed to be run with continuous user input and disconnect from the terminal when task completed. For example, a web server responds to web requests, rather than user input. Mail servers are another example of this type of application. These types of programs are known as daemons. The term daemon comes from Greek mythology and represents an entity that is neither good nor evil, and which invisibly performs useful tasks.

Daemons are Disk and Execution Monitor. A daemon is a long-running background process that answers requests for services. The term originated with Unix, but most operating systems use daemons in some form or another. In Windows platform for example, daemons are called "services". In Unix, the names of daemons conventionally end in "d". Some examples include inetd, httpd, nfsd, sshd, named, and lpd.

A daemon process are not attached to any terminal. So in "ps -ef" command output all daemons will have a '?' the tty field. Most daemons be owned by root.

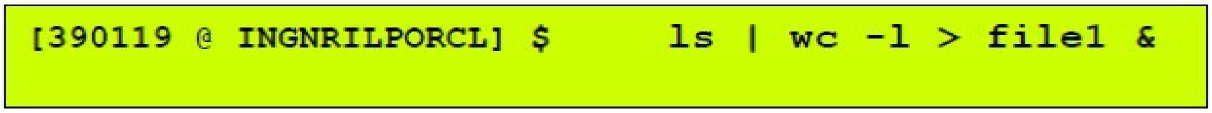
## *1.5 Starting a Process :*

Process can be started in two ways:

**In Foreground :** By default every process starts in foreground, ie. Gets the input from keyboard and sends the output in monitor. But in this case till the process completes its execution no other process can be started in foreground.

**In Background :** To take the advantage multiprocessing environment, a process can be started in background, so that other process can be started in the foreground without waiting for the previous process to complete execution.

A process can be started in background by adding ampersand(&) after it.



```
[390119 @ INGNRILPORCL] $     ls | wc -l > file1 &
```

## *1.6 Switching process from foreground to background*

A process running in foreground can be send to background using the following steps:

>> Press <ctrl> + Z to suspend the job

>> bg command puts the job in background

>> nohup <PID> unattaches the job from the terminal

```
$  cat > file
File content...
<CTRL> + z
[1] + Stopped
$ bg
[1] + cat > file &
```

## 1.7 Switching process from background to foreground

Process running in the background can be taken into foreground using thefollowing steps:

>> Find the job id of the process by the command jobs

>> Use fg %<jobid>to get the job to foreground

```
$  jobs
[1] - Sttoped cat > file

$  fg %1
cat > file

Some more content...
<ctrl> + d

$ cat file
File content...
Some more content...
$
```

## *1.8 Stopping/Killing a Process*

A process dies(terminates ) automatically when it completes the job it

intended to.

A process can be killed abnormally with the command kill.

>> Use the ps command to find out the process-id for the process

>> Use the command kill to terminate it

>> Use the command kill -9 to kill a process forcefully.

Only the creator of the process or the super user root has the required permission to kill a process. One exception, even root cannot kill a process with process-id 1.

```
$  ps
PID          TTY      TIME         CMD
15982        tty4     00:00:00     sh
18700        tty4     00:02:35     cat
18746        tty4     00:00:00     ps
$ kill 18700
Terminated

$ ps
PID          TTY      TIME         CMD
15982        tty4     00:00:00     sh
18746        tty4     00:00:00     ps
$
```

A kill command sends a SIGTERM signal to the identified process, which allows the process to run a proper shutdown routine, where as a kill -9 delivers a SIGKILL signal that stops and kills the process immediately (which could lead to data loss or crash in some cases).

## 1.9 Option settings for background process

The effect of logging off( hung-up) on currently running process:

While a process is executing/running , if the owner tries to log off the process command will get killed, on the other hand sometimes a job or command takes a long time to complete. If it is required the job to be completed without interruption.

In unix the situation can be handled in two different ways:

>> Does not allow the use to log off

>> Continue execution in background even after the user logged off This can be achieved using command nohup