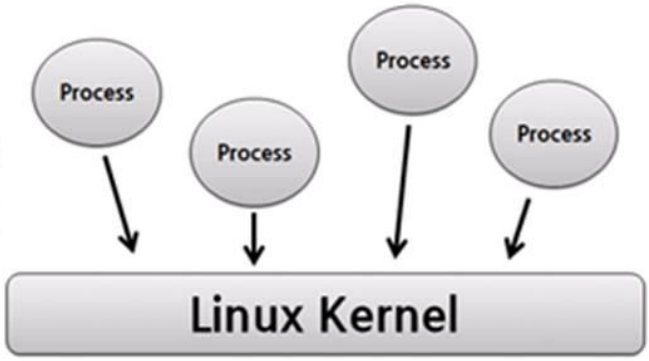
MANAGING RUNNING PROCESS



What is a Process

- A process is a running instance of a command.
- A process is identified on the system by what is referred to as a process ID.
- That process ID is unique for the current system.
- The ability to manage processes on your system is critical for a Linux system administrator.

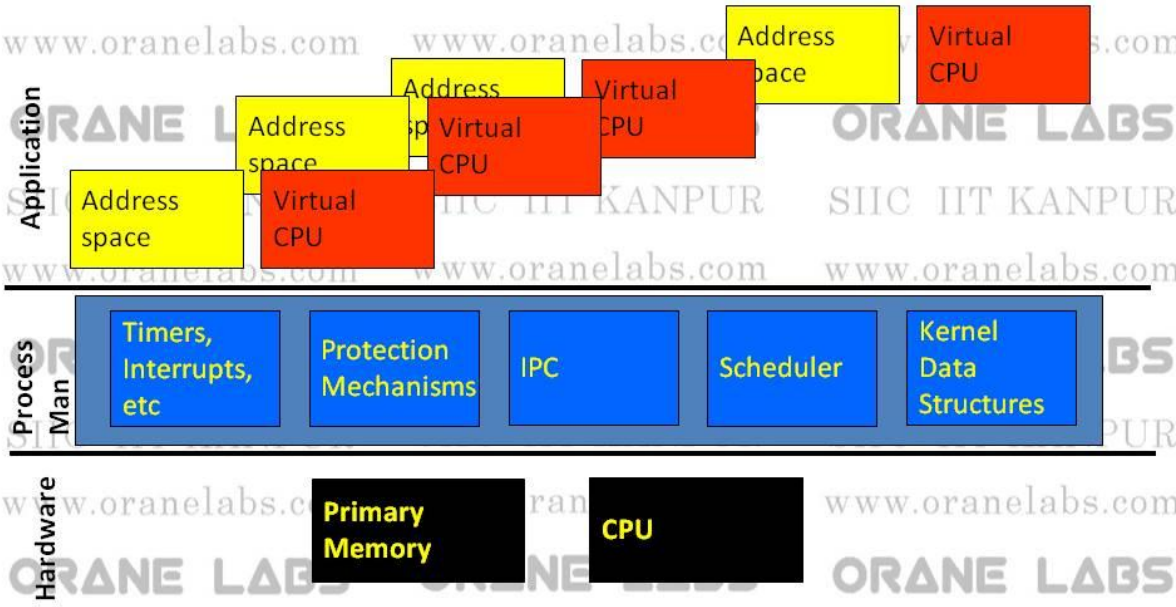
Kernel View of Process

- The kernel's internal data structures record various pieces of information about each process. Here are some of the more important of these:
  - The process's address space map
  - The current status of the process (sleeping, stopped, runnable, etc.)
  - The execution priority of the process
  - Information about the resources the process has used
  - Information about the files and network ports the process has opened
  - The process's signal mask (a record of which signals are blocked)
  - The owner of the process

Process States

- **Running :** The process is either running (it is the current process in the system) or it is ready to run (it is waiting to be assigned to one of the system's CPUs).
- **Waiting :** The process is waiting for an event or for a resource. Linux differentiates between two types of waiting process; *interruptible* and *uninterruptible*.
- **Stopped :** The process has been stopped, usually by receiving a signal. A process that is being debugged can be in a stopped state.
- **Zombie :** This is a halted process which, for some reason, still has a task_struct data structure in the task vector. It is what it sounds like, a dead process.

Process Management



Process Tracking

- It is easy to see your own processes by running the ps(process status) command. Issued without any arguments, ps displays your own processes that are associated with a terminal.

```
$ ps
  PID TTY          TIME CMD
18358 ttyp3    00:00:00 sh
18361 ttyp3    00:01:31 abiword
18789 ttyp3    00:00:00 ps
```

Process States

- To track the process state, use the following command:
  - ps u

```
$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
reed     18358  0.0  0.7   2460   668 ttyp3    S    Sep13   0:00 -sh
reed     18361  0.0 11.6 32936 10756 ttyp3    S    Sep13   6:12 abiword
reed     19736  0.0  0.8   2508   784 ttyp3    R    18:12   0:00 ps u
```

✓STAT- S means Sleeping and R Means Running State.
✓a plus sign (+) indicates that the process is associated with the foreground operations.

More.....

- VSZ (virtual set size) shows the size of the image process (in kilobytes),
- RSS (resident set size) shows the size of the program in memory.
- The VSZ and RSS sizes may be different because VSZ is the amount of memory allocated for the process, whereas RSS is the amount that is actually being used.
- START shows the time the process began running.
- TIME shows the cumulative system time used.

Paging through Processes

- Many processes can be running on a system.
- Sometimes it is useful to page through the process.
- This can be done by pipelining the ps output to the less command.
  - $ ps ux | less
- To page through all processes running for all users on your system, use the ps aux command as follows:
  - $ ps aux | less

Formatting Output

- The ps command can be customized to display output in columnar format.
- You can display information of selected columns.
- You can also sort the output by columns.
- All this can be done using the –o option.

Example

- $ ps -eo 'pid,user,uid,group,gid,vsz,rss,comm' | less
- PID USER GROUP GID VSZ RSS COMMAND

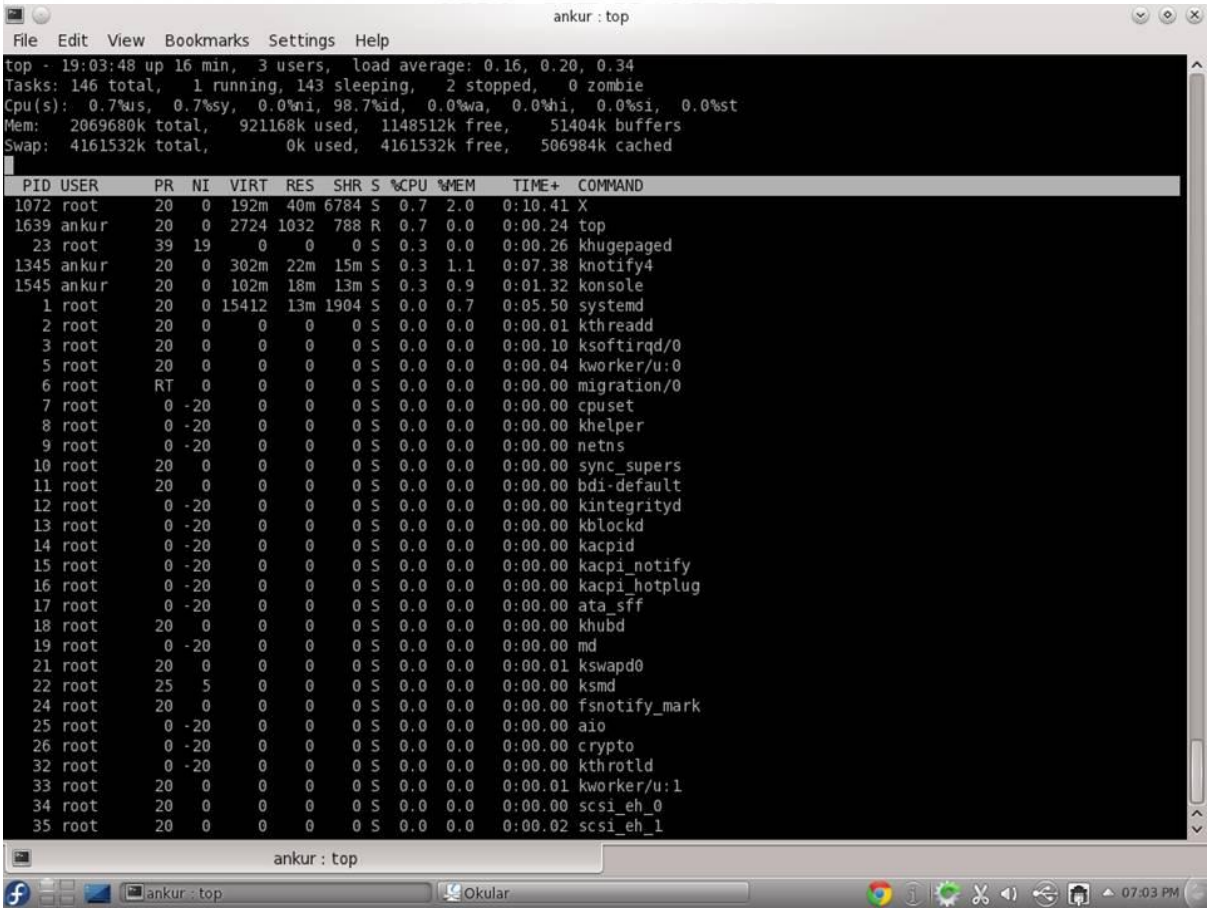
```
  1 root     root           0  19324  1320 init
  2 root     root           0      0     0 kthreadd
```


Sorting Process

- If you want to sort by a specific column, you can use the sort= option.
- For example, to see which processes are using the most memory, I sort by the rss field
  - $ ps -eo 'pid,user,group,gid,vsz,rss,comm' --sort=-rss | less

Listing and changing processes with top

- The top command displays processes running on your system.
- With top, the default is to display processes based on how much CPU time they are currently consuming.
- You can also use top to kill processes. But can be done only by becoming a root user.



Information about your system

- General information about your system appears at the top of the top output.
- Followed by information about each process.
- At the top, you can see how long the system has been up, how many users are currently logged in.
- and how much demand there has been on the system for the past 1, 5, and 10 minutes.

Other options with top

- **Press h** to see help options, and then press any key to return to the top display.
- **Press M** to sort by memory usage instead of CPU, and then press P to return to sorting by CPU.
- **Press the number 1** to toggle showing CPU usage of all your CPUs, if you have more than one CPU on your system.
- **Press R** to reverse sort your output.
- **Press u** and enter a username to display processes only for a particular user.

System Processes

- To see information about System process the following command can be used:
  - ps ax
- The total number of system process can found out using the wc –l command. Example:
  - ps ax | wc –l

The Process Tree

- A process tree displays the lineage of your different processes, placing a child process with its parent.
  - pstree

Managing Background and Foreground Processes

- You can selectively manage process to run in the background or foreground.
- To stop a running command and put it in the background, press Ctrl+Z.
- After the command is stopped, you can either bring it back into the foreground to run (the **fg command**) or start it running in the background (the **bg command**).

Starting background processes

- To place a program in the background at the time you run the program, type an ampersand (&) at the end of the command line, like this:
  - $ find /usr > /tmp/allusrfiles &
  - [3] 15971
- To check which commands you have running in the background, use the jobs command, as follows:
  - $ jobs

Using foreground and background commands

- Using the fg command you can bring back a command to the foreground.
  - fg %
- You can also use the following to refer to a background job:
  - **%** — Refers to the most recent command put into the background (indicated by the plus sign when you type the jobs command). This action brings the command to the foreground.
  - **%string** — Refers to a job where the command begins with a particular string of Characters
  - **%?string** — Refers to a job where the command line contains a string at any point.
  - **%--** — Refers to the previous job stopped before the one most recently stopped.

- If a command is stopped, you can start it running again in the background using the bg command. For example, take job 5 from the jobs list in the previous example:
  - [5]+ Stopped
- Type the following:
  - $ bg %5

Killing and Renicing Processes

- kill and killall commands are used for killing processes manually.
- Process are killed by sending signals.
- Signals are represented by both numbers and names.

Available Signals in Linux

Signal	Number	Description
SIGHUP	1	Hangup detected on controlling terminal or death of controlling process.
SIGINT	2	Interrupt from keyboard.
SIGQUIT	3	Quit from keyboard.
SIGABRT	6	Abort signal from abort(3).
SIGKILL	9	Kill signal.
SIGTERM	15	Termination signal.
SIGCONT	19,18,25	Continue if stopped.
SIGSTOP	17,19,23	Stop process.

Using kill to signal processes by PID

- here are some examples of the kill command you could use to kill that process:
  - $ kill 10432
  - $ kill -15 10432
  - $ kill -SIGKILL 10432
- To kill a bunch of processes with same name use the killall command.

Setting processor priority with nice and renice

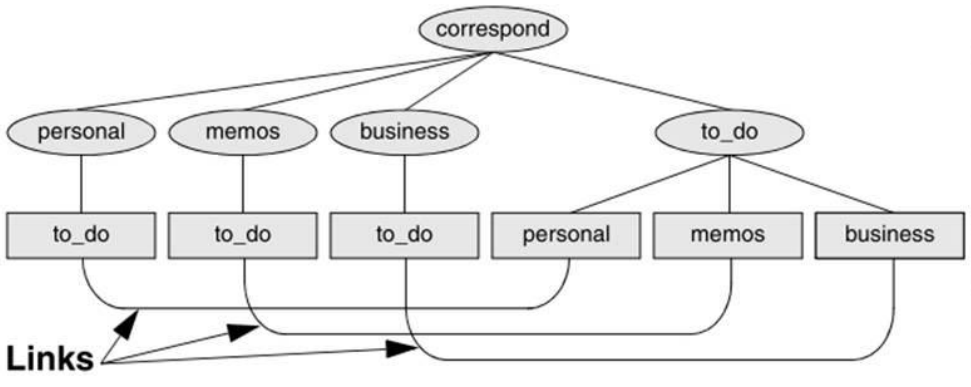
- Every process running on your system has a nice value between –20 and 19.
- By default, the nice value is set to 0.
- Here are a few facts about nice values:
  - The lower the nice value, the more access to the CPUs the process will have.
  - A regular user can set nice values only from 0 to 19. No negative values are allowed.
  - A regular user can set the nice value higher, not lower
  - A regular user can set the nice value only on the user's own processes.
  - The root user can set the nice value on any process to any valid value, up or down.

The nice and renice command

- You can use the nice command to run a command with a particular nice value.
- Example:
  - nice +5 updatedb &
- renice command:
  - renice -n -5 20284

Links

- A link is a pointer to a file.
- This pointer associates a filename with a place on the disk.



Types of Link

- There are two different types of file links in Linux:
  - A symbolic, or soft, link
  - A hard link

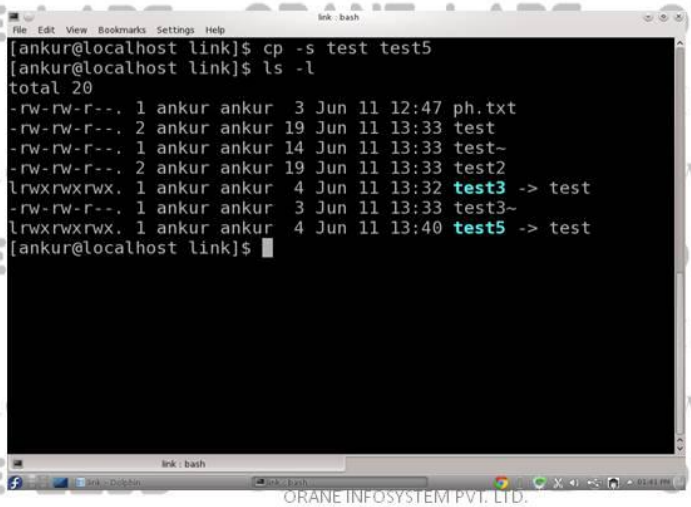
Hard Link

- The hard link creates a separate file that contains information about the original file and where to locate it.
- When you reference the hard link file, it's just as if you're referencing the original file.

```
link : bash
File  Edit  View  Bookmarks  Settings  Help
[ankur@localhost ~]$ cd link
[ankur@localhost link]$ cat >test
hello, this is a link test
^Z
[1]+  Stopped                 cat > test
[ankur@localhost link]$ cp -l test test2
[ankur@localhost link]$ ls -il
total 12
1058128 -rw-rw-r--. 1 ankur ankur  3 Jun 11 12:47 ph.txt
1058129 -rw-rw-r--. 2 ankur ankur 27 Jun 11 13:20 test
1058129 -rw-rw-r--. 2 ankur ankur 27 Jun 11 13:20 test2
[ankur@localhost link]$
```


Soft Link

- On the other hand, the -s parameter creates a symbolic, or soft, link:



Points to remember

- A hard link always refers to a file within the same file system.
- A soft link can refer to files in other file systems as well.
- Modification to any of the files will be visible in all the files.
- A hard linked file will have the same inode number as the orignal file.
- A soft linked file will have a different inode number.

Compressing data

Utility	File extension	Description
bzip2	.bz2	Uses the Burrows-Wheeler block sorting text compression algorithm and Huffman coding
compress	.Z	Original Unix file compression utility; starting to fade away into obscurity
gzip	.gz	The GNU Project's compression utility; uses Lempel-Ziv coding
zip	.zip	The Unix version of the PKZIP program for Windows

bzip2

- By default, the bzip2 command attempts to compress the original file, and replaces it with the compressed file, using the same file name with a .bz2 extension:
  - $ ls -l myprog
    - -rwxrwxr-x 1 rich rich 4882 2007-09-13 11:29 myprog
  - $ bzip2 myprog
  - $ ls -l my*
    - -rwxrwxr-x 1 rich rich 2378 2007-09-13 11:29 myprog.bz2
  - $

Uncompressing

- To uncompress the file, just use the bunzip2 command:
  - $ bunzip2 myprog.bz2
  - $ ls -l myprog
    - -rwxrwxr-x 1 rich rich 4882 2007-09-13 11:29 myprog
  - $

The gzip utility

- By far the most popular file compression utility in Linux is the gzip utility. The gzip package is a creation of the GNU Project.
- This package includes the files:
  - gzip for compressing files
  - gzcat for displaying the contents of compressed text files
  - gunzip for uncompressing files.

- These utilities work the same way as the bzip2 utilities:
  - $ gzip myprog
- The gzip command compresses the file you specify on the command line.
  - $ gzip my*

THE VI EDITOR

Starting the vi Editor

- vi editor can started from the shell by issuing the following command:
  - vi file_name
- If this is a new file, you should see something similar to the following:

Two Modes

- First, you need to know the two main operating modes:
  - command and
  - input.
- The vi editor always starts in command mode.
- To enter text you have be in the input mode.

Adding Text

- **a**—The add command. With this command, you can input text that starts to the right of the cursor.
- **A**—Add at end command. With this command, you can input text starting at the end of the current line.
- **i**—The insert command. With this command, you can input text that starts to the left of the cursor.
- **I**—The insert at beginning command. With this command, you can input text that starts at the beginning of the current line.
- **o**—The open below command. This command opens a line below the current line and puts you in insert mode.
- **O**—The open above command. This command opens a line above the current line and puts you in insert mode.

Moving around in the text

- Arrow keys—Move the cursor up, down, left, or right in the fi le one character at a time. To move left and right, you can also use Backspace and the spacebar, respectively. If you prefer to keep your fi ngers on the keyboard, move the cursor with h (left), l (right), j (down), or k (up).
- w—Moves the cursor to the beginning of the next word (delimited by spaces, tabs, or punctuation).
- W—Moves the cursor to the beginning of the next word (delimited by spaces or tabs).
- b—Moves the cursor to the beginning of the previous word (delimited by spaces, tabs, or punctuation).
- B—Moves the cursor to the beginning of the previous word (delimited by spaces or tabs).
- 0 (zero)—Moves the cursor to the beginning of the current line.
- $—Moves the cursor to the end of the current line.

Deleting, copying, and changing text

- The x, d, y, and c commands can be used to delete and change text.
  - x—Deletes the character under the cursor.
  - X—Deletes the character directly before the cursor.
  - d<?>—Deletes some text.
  - c<?>—Changes some text.
  - y<?>—copies some text.

Other Options

- dw—Deletes (d) a word (w) after the current cursor position.
- db—Deletes (d) a word (b) before the current cursor position.
- dd—Deletes (d) the entire current line (d).
- c$—Changes (c) the characters from the current character to the end of the cur-rent line ($) and goes into input mode.
- c0—Changes (c) from the previous character to the beginning of the current line (0) and goes into input mode.
- cl—Erases (c) the current letter (l) and goes into input mode.
- cc—Erases (c) the line (c) and goes into input mode.
- yy—Copies (y) the current line (y) into the buffer.
- y)—Copies (y) the current sentence ( )), to the right of the cursor, into the buffer.
- y}—Copies (y) the current paragraph (}), to the right of the cursor, into the buffer.

Pasting (putting) text

- P—Puts the copied text to the left of the cursor if it's letters or words; puts the copied text above the current line if it contains lines of text.
- p—Puts the buffered text to the right of the cursor if it's letters or words; puts the buffered text below the current line if it contains lines of text.

Exiting vi

- ZZ—Save the current changes to the file and exit from vi.
- :w—Save the current file but continue editing.
- :wq—Same as ZZ.
- :q—Quit the current file. This works only if you don't have any unsaved changes.
- :q!—Quit the current file and don't save the changes you just made to the file.

Searching for text

- To search for the next or previous occurrence of text in the file, use either the slash (/) or the question mark (?) character.
  - /hello—Searches forward for the word hello.
  - ?goodbye—Searches backward for the word goodbye.
  - /The.*foot—Searches forward for a line that has the word The in it and also, after that at some point, the word foot.
  - ?[pP]rint—Searches backward for either print or Print.
  - After you have entered a search term, simply type n or N to search forward or backward for the same term again, respectively.

ENVIRONMENT VARIABLES

What are They?

- The bash shell uses a feature called environment variables to store information about the shell session and the working environment.
- There are two types of environment variables in the bash shell:
  - Global variables
  - Local variables

Global environment variables

- Global environment variables are visible to all the processes.
- Viewing the global Variables:
  - $ printenv
- To display the value of an individual environment variable, use the echo command.
  - echo $HOME

Local Variables

- These variables are local to the process in which they are declared.
- Viewing can be tricky.
- Use the set command to view all the variables declared.

Setting Environment Variables

- **Setting local environment variables**
- You can assign either a numeric or a string value to an environment variable by assigning the variable to a value using the equal sign:
  - $ test=testing
  - $ echo $test
  - testing
  - $
- If you need to assign a string value that contains spaces, you'll need to use a single quotation mark to delineate the beginning and the end of the string:
  - $ test=testing a long string
  - -bash: a: command not found
  - $ test='testing a long string'
  - $ echo $test
  - testing a long string
  - $

Setting global environment variables

- The method used to create a global environment variable is to create a local environment variable, then export it to the global environment. This is done by using the export command:
  - $ echo $test
  - testing a long string
  - $ export test
  - $ bash
  - $ echo $test
  - testing a long string
  - $

Removing Environment Variables

- This is done by using the unset command:
  - $ echo $test
  - testing
  - $ unset test
  - $ echo $test
  - $

Default Variables

Variable	Description
CDPATH	A colon-separated list of directories used as a search path for the cd command.
HOME	The current user's home directory.
IFS	A list of characters that separate fields used by the shell to split text strings.
MAIL	The filename for the current user's mailbox. The bash shell checks this file for new mail.
MAILPATH	A colon-separated list of multiple filenames for the current user's mailbox. The bash shell checks each file in this list for new mail.
OPTARG	The value of the last option argument processed by the getopts command.
OPTIND	The index value of the last option argument processed by the getopts command.
PATH	A colon-separated list of directories where the shell looks for commands.
PS1	The primary shell command line interface prompt string.
PS2	The secondary shell command line interface prompt string.

BASH Variables

BASH_VERSION	A number representing the current version of the bash command.
HOME	This is your home directory.
HOSTTYPE	A value that describes the computer architecture
PATH	The colon-separated list of directories used to fi nd commands
PPID	The process ID of the command that started the current shell
PWD	This is the directory that is assigned as your current directory

Setting the PATH Environment Variable

- You can add new search directories to the existing PATH environment variable without having to rebuild it from scratch.
- The individual directories listed in the PATH are separated by a colon.

- You can view the $PATH variable as:
  - echo $PATH
- For adding to the $PATH variable:
  - $ PATH=$PATH:/home/ankur/test