



**RHSA1**  
**Red Hat System Administration I**  
**Day 4**

# Day 4 Contents

- Processes, priorities and signals Concepts .
- Redirection.
- Pipe Line.
- Word Count.



# Processes

---

- Every program you run creates a process.

## Example:

- Shell
- Command
- An application

# Processes

---

- For everything that happens on a Linux server, a process is started.
- System starts processes called daemons which are processes that run in the background and provide services.
- Every processes has a PID.
- When a process creates another, the first is the **parent** of the new process. The new process is called the **child process**.

# Processes

---

- Types of process:
  - Shell jobs.
  - Daemons.
  - Kernel threads.

# Viewing Processes

- Ps (process status) command

ps [options]

- Output

- PID.
- TTY -> terminal identifier.
- Execution time.
- Command name.

- Options

- -e: all system processes.
- -f: full information.
- -u uid: display processes of that user.
- a: all processes attached to a terminal.
- x: all other processes.

# Shell jobs

- Shell jobs are commands started from the command line. They are associated with the shell that was current when the process was started.
- When a user types a command, a shell job is started.
- By default, any executed command is started as **foreground** job.
- If you know that a job will take a long time to complete, you can start it in the background with an **&** behind it.
- This immediately starts the job in the **background** to make room for other tasks to be started from the command line.

# Shell jobs

- A signal is a message sent to a process to perform a certain action.
- To send signals to processes or process group, you can use kill command, killall command or pkill command.
- Kill **-[signal]** PID `kill 12047`
- Pkill **-[signal]** process\_name `kill -9 mail`
- Killall process\_name `killall vim`



# Shell jobs

- Signals are identified by a signal number and a signal name, and has an associated action.
  - SIGTERM → 15
  - SIGKILL → 9
- If no signal is specified, the TERM signal is sent.
- For complete overview of all the available signals, you can use **man 7 signal**.

# Shell jobs

## Examples

- `sleep 3600&`  
`[1] 3302`
- `jobs`  
`[1]+ Running sleep 3600 &`
- `fg 1`  
`sleep 3600`
- `ctrl+z` `[1]+ Stopped sleep 3600`
- `jobs`  
`[1]+ Stopped sleep 3600`
- `bg %1`  
`[1]+ sleep 3600 &`

# Shell jobs

## Examples

- jobs  
[1]+ Running sleep 3600 &
- kill -SIGSTOP %1  
[1]+ Stopped sleep 3600
- kill %1  
[1]+ Terminated sleep 3600
- jobs

# Processes

- Every process has a **parent process**, and as long as it lives, the parent process is responsible for the **child processes** it has created.
- **In older versions of Linux**, killing a parent process would kill all of its child processes.
- **In RHEL 8**, if you kill a parent process, all of its child processes become children of the systemd process.

# Processes Priority

- When Linux processes are started, they are started with a specific priority.
- By default, all regular processes are equal and are started with the same priority, which is the priority number 20.
- Every process which is ready to run has a scheduling priority.
- The Linux process divides CPU time into time slices, in which each process will get a turn to run, higher priority processes first.
- User can affect the priority by setting the niceness value for a process.

# Adjusting Priority

- Niceness values range from -20 to +19, which indicates how much of a bonus or penalty to assign to the priority of the process.
- To change the default priority that was assigned to the process when it was started.
- Use **nice** if you want to start a process with an adjusted priority.
- **nice** [-n adjustment] command  
`nice -n 5 dd if=/dev/zero of=/dev/null &`

# Adjusting Priority

- Use **renice** to change the priority for a currently active process, or you can use the **r** command from the **top utility** to change the priority of a currently running process.
- **renice** priority **[[ -p ] pid ...]** **[[ -g ] group ...]** **[[ -u ] user ...]**  
  
ps | aux  
  
renice -n 10 -p 1234
- The default niceness of a process is set to 0 (which results in the priority value of 20).

# Adjusting Priority

- By applying a negative niceness, you increase the priority.
- Use a positive niceness to decrease the priority.
- Do not set process priority to -20, it risks blocking other processes from getting served.
- The regular users can only decrease the priority of a running process.
- You must be root to give processes increased priority.



# Viewing Processes

---

- Use **top** to display Linux processes.
- The top program provides a dynamic real-time view of a running system.
- It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel.

# Choose

1. Which of the following is not generally considered a type of process?  
(Choose two.)
  - a. A shell job
  - b. A cron job
  - c. A daemon
  - d. A thread
  
2. Which of the following can be used to move a job to the background?
  - a. Press &
  - b. Press Ctrl-Z and then type **bg**
  - c. Press Ctrl-D and then type **bg**
  - d. Press Ctrl-Z, followed by &

# Choose

5. Which of the following commands is most appropriate if you're looking for detailed information about the command and how it was started?
  - a. `ps ef`
  - b. `ps aux`
  - c. `ps`
  - d. `ps fax`
6. Of the following **nice** values, which will increase the priority of the selected process?
  - a. 100
  - b. 20
  - c. -19
  - d. -100
7. Which of the following shows correct syntax to change the priority for the current process with PID 1234?
  - a. `nice -n 5 1234`
  - b. `renice 5 1234`
  - c. `renice 5 -p 1234`
  - d. `nice 5 -p 1234`



# Choose

8. Which of the following commands cannot be used to send signals to processes?
  - a. `kill`
  - b. `mkill`
  - c. `pkill`
  - d. `killall`
9. Which of the following commands would you use from **top** to change the priority of a process?
  - a. `r`
  - b. `n`
  - c. `c`
  - d. `k`

# Choose

---

1. Which command gives an overview of all current shell jobs?
2. How do you stop the current shell job to continue running it in the background?
3. Which keystroke combination can you use to cancel the current shell job?
4. A user is asking you to cancel one of the jobs he has started. You cannot access the shell that user currently is working from. What can you do to cancel his job anyway?

# Choose

5. Which command would you use to show parent-child relationships between processes?
6. Which command enables you to change the priority of PID 1234 to a higher priority?
7. On your system, 20 **dd** processes are currently running. What is the easiest way to stop all of them?
8. Which command enables you to stop the command with the name **mycommand**?
9. Which command do you use from **top** to kill a process?
10. What is required to select a performance profile that best matches your system needs?

# Searching For A Process

- To search for a process, you can use **pgrep** command.
- **pgrep option(s) pattern.**
- Options
  - x: exact match.
  - u uid: processes for a specific user.
  - l: display the name with pid.

# Standard Input And Output

- **Standard input:**
  - Refers to the data source from which data is input to a command.
  - Typically the keyboard.
- **Standard output:**
  - Refer to data destination to which data from the command is written.
  - Typically the screen.
- **Standard error:**
  - Refer to the output destination for the errors and messages generated by the command.
  - Typically the screen also.



# Standard Input And Output

- In I/O redirection, files can be used to replace the default **standard input**, **standard output** and **standard error**.
- You can also redirect to device files.
- If you want to discard a command's output, you can redirect to **/dev/null**.

# Redirection

- **Standard input:**
  - `command < fname`
- **Standard output:**
  - `command > fname`
  - `command >> fname`
- **Standard error:**
  - `command 2> fname`
  - `command 2> /dev/null`

# Redirection

- Examples:

`ls -R / > file.txt 2> /dev/null`

`ls -l /etc >> findresult`

`find /etc -name passwd > findresult`

`find / -name passwd 2> errs > results`

`mail < file2.txt`

`sort < file2.txt > sortedf1.txt`

# Pipe Line

- A pipe (|) is used to send the output of one command as the input to another.
- Command 1 | Command 2.
- Examples:

ls -lR | more

ps -ef | more

history | more

# The tee Command

---

- The tee command reads from the standard input and writes to the standard output and a file.

- Examples:

ls -lR / | tee fname | more

# String Processing

- Use the **wc** and the **diff** commands to gather word file statistics and compare two files.
- Search strings for patterns using the **grep** command.
- Move and delete data using **cut** and **paste** commands.
- Organize data using the **sort**, and **paste** command.

# The **wc** command

- The **wc** command displays the number of characters, words, and lines in a specified file.
- The syntax for the **wc** command is:  
**wc [option] [filename]**
- The **wc** command is often used when differentiating between two versions of a file.

# The **wc** command

- Word-count command options

Option	Meanings
-c	Count the number of characters only.
-l	Count the number of lines only.
-w	Counts the number of words only

- Example:

**wc** story.txt

39 237 1901 story.txt



# The diff command

- The diff command is also used to compare the contents of two files for differences. If you upgrade a utility and want to see how the new configuration files differ from the old, use the diff command.
- `diff /etc/named.conf.rpm.new /etc/named.conf`

will give the output as:

```
20c20
```

```
<
```

```
---- file "root.hints";
```

```
> file "named.ca"
```

# The grep command

- Displays the lines of its input that match a pattern given as an argument.
- The syntax for the grep command is:

**grep [options] regular-expression filename(s)**

Option	Description
-i	Not case sensitive.
-v	Only shows lines that do not contain the regular expression.
-r	Searches files in the current directory and all subdirectories.

# The tr command

- The tr command can be used to translate characters from standard input and write to standard output.
- The syntax for the tr command is:  
`tr [option] string1 string2`
- Example
- `echo "Hello, world." | tr 'a-z' 'A-Z'`  
`HELLO, WORLD`

# The cut command

- cut command cuts fields or columns of text from standard input or the named file and displays the result to standard.

- The syntax for the cut command is:

`cut option[s] [filename]`

- Options

- -f specifies field or column.
- -d specifies field delimiter (default is TAB).
- -c specifies characters and cuts by characters.

- Example

```
cut -f3 -d: /etc/passwd
```

# The sort command

- The sort command sorts text data after accepting it from either a file or the output of another command.
- The sorted text is sent to the standard output, with the original file remaining unchanged in the process.
- The syntax for the sort command is:

`sort option[s] [filename]`

- Example

```
sort -t : -k1 /etc/passwd
```

```
sort -t : -k3 /etc/passwd
```

```
sort -t : -n -k3 -o passwd_sorted /etc/passwd
```