


# Linux System Administration



## LINUX Distributions

- ❏ A combination of a kernel and all sorts of software, tools and choices into one bundle is called as **distribution**.
- ❏ Distributions have their definite look-and-feel, their culture, their fans & enemies.
- ❏ Kernel and software might be the thread and fabric, but the Distribution is the suit or dress and that should fit you !

## File System

- 
- ❏ The Unix file system looks like an inverted tree structure.
  - ❏ You start with the root directory, denoted by /, at the top and work down through sub-directories underneath it.

## File System

- Each node is either a file or a directory of files, where the latter can contain other files and directories.
- You specify a file or directory by its path name, either the full, or absolute, path name or the one relative to a location.
- The full path name starts with the root, /, and follows the branches of the file system, each separated by /, until you reach the desired file, e.g.:
  - `/home/condron/source/xntp`

# Users, Groups and Access Permissions



- ❑ In UNIX/LINUX, there is a concept of user and an associated group
- ❑ The system determines whether or not a user or group can access a file or program based on the permissions assigned to them.
- ❑ Apart from all the users, there is a special user called Super User or the root which has permission to access any file and directory

## Access Permissions



---

There are three permissions for any file, directory or application program.

The following lists the symbols used to denote each, along with a brief description:

**r** — Indicates that a given category of user can read a file.

**w** — Indicates that a given category of user can write to a file.

**x** — Indicates that a given category of user can execute the file.

## Access Permissions

- Each of the three permissions are assigned to three defined categories of users.
- The categories are:
  - owner* — The owner of the file or application.
  - group* — The group that owns the file or application.
  - others* — All users with access to the system.

The three types of Linux support provided by Red Hat are as follows:

**Hardware Support** Red Hat has agreements with every major server hardware vendor to

- make sure that whatever server a customer buys, the hardware vendor will assist them in fixing hardware issues, when Red Hat is installed on it.

**Software Support** Red Hat has agreements with every major enterprise software vendor to

- make sure that their software runs properly on top of the Red Hat Linux operating system
- and that the enterprise software is also guaranteed to run on Red Hat Linux by the vendor of the operating system.

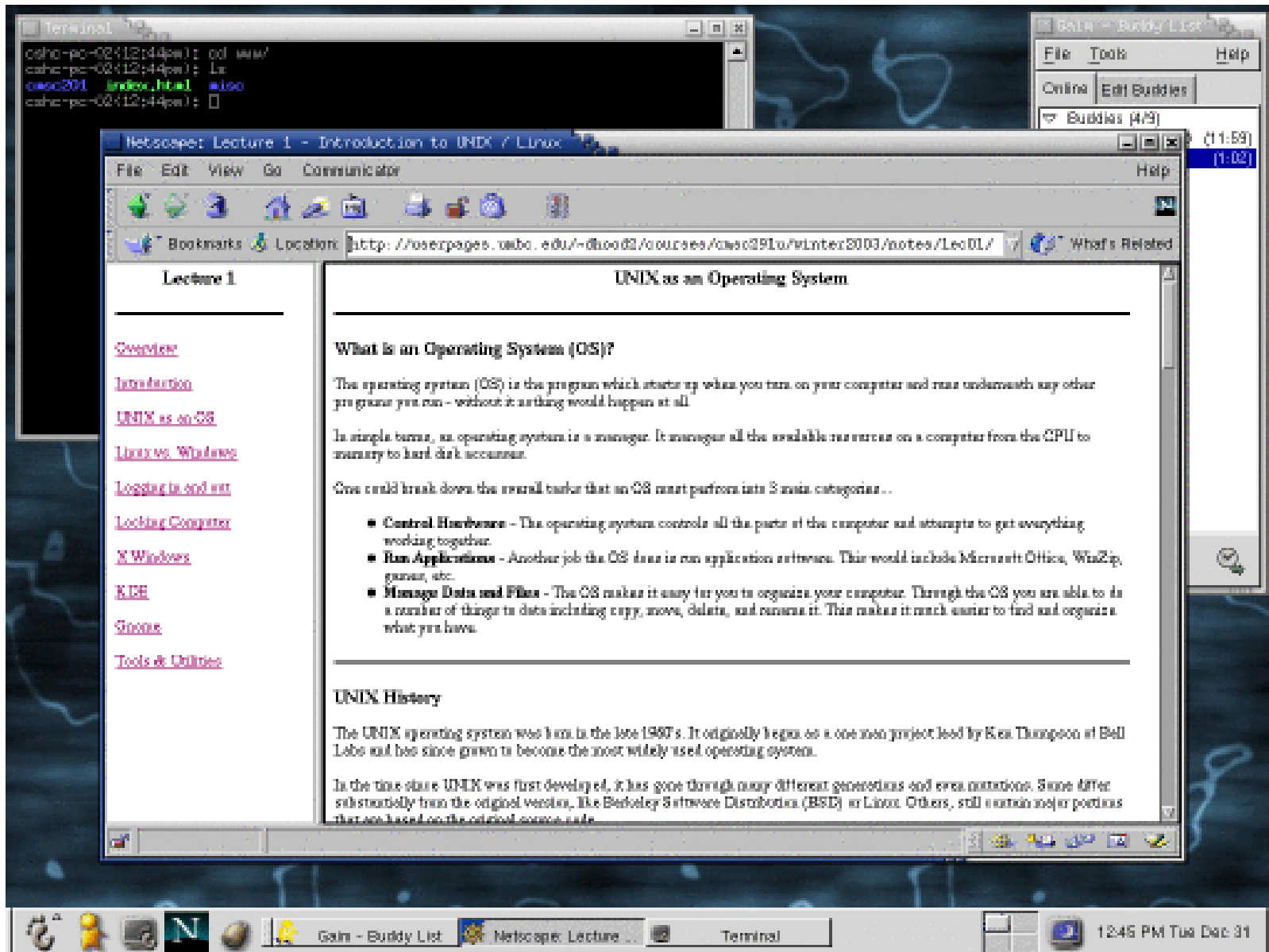
**Hands-on Support** This means that if a customer is experiencing problems accomplishing tasks with Red Hat software, the Red Hat Global Support organization is there to help them by fixing bugs and providing technical assistance.



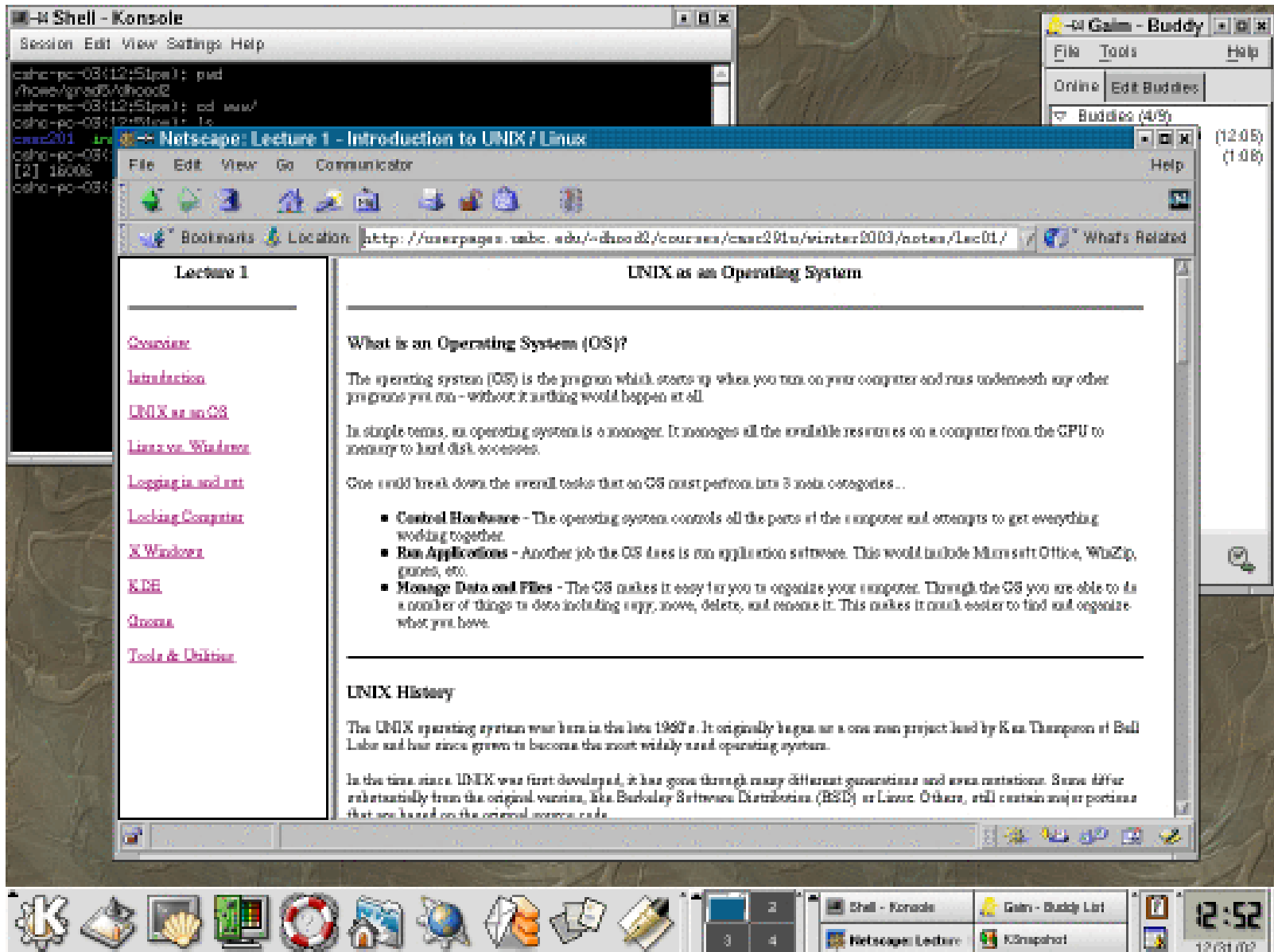
# Graphical User Interfaces (GUIs)

- When you logon locally, you are presented with graphical environment.
- You start at a graphical login screen. You must enter your username and password. You also have the option to choose from a couple session types. Mainly you have the choice between Gnome and KDE.
- Once you enter in your username and password, you are then presented with a graphical environment that looks like one of the following...

# Gnome



# KDE



# Bourne shell

- The Bourne shell is the original UNIX shell (command execution program, often called a command interpreter) that was developed at AT&T.
- Named for its developer, Stephen Bourne, the Bourne shell is also known by its program name, sh.
- The shell prompt (character displayed to indicate readiness for input) used is the \$ symbol.
- The Bourne shell family includes the Bourne, Korn shell, bash, and zsh shells.

# Working with the Bash Shell

- To communicate commands to the operating system kernel, an interface is needed that sits between the kernel and the end user issuing these commands.
- This interface is known as the shell.
- Several shells are available on RHEL.
- Bash (short for the Bourne Again Shell) is the one that is used in most situations. This is because it is compatible with the Bourne shell, which is commonly found on UNIX servers.

# Getting the Best of Bash

- Bash has some useful features to offer.
- Some of the most used Bash features are **automatic completion** and the **history mechanism**.
- To open the file **this\_is\_a\_file**, the user can type **cat thi** and then immediately hit the Tab key.
- If there is just one file that starts with the letters thi, Bash will automatically complete the name of the file.
- If there are more options, Bash will complete the name of the file as far as possible.

# Useful Bash Key Sequences

- Sometimes, you will enter a command from the Bash command line and nothing, or something totally unexpected, will happen.
- **Ctrl+C** Use this key sequence to quit a command that is not responding (or simply is taking too long to complete).
- **Ctrl+D** This key sequence is used to send the end-of-file (EOF) signal to a command. Use this when the command is waiting for more input. It will indicate this by displaying the secondary prompt `>`.

- **Ctrl+R** This is the reverse search feature. This feature helps you locate commands you have used previously. The feature is especially useful when working with longer commands. Type the first characters of the command, and you will immediately see the last command you used that started with the same characters.
- **Ctrl+Z** A command that is interrupted with Ctrl+Z is just halted until it is started again with the fg command as a foreground job or with the bg command as a background job.
- **Ctrl+A** The Ctrl+A keystroke brings the cursor to the beginning of the current command line.
- **Ctrl+B** The Ctrl+B keystroke moves the cursor to the end of the current command line.



# Piping and Redirection

- most powerful features of the Linux command line.
- **Piping** is used to send the result of a **command to another command**.
- **Redirection** sends the output of a **command to a file**.
- This file doesn't necessarily need to be a regular file, but it can also be a device file.

## Discovering the Use of Pipes

In this exercise, you'll see how a pipe is used to add functionality to a command. First you'll execute a command where the output doesn't fit on the screen. Next, by piping this output through `less`, you can see the output screen by screen.

1. Open a shell, and use `su -` to become the root. Enter the root password when prompted.
2. Type the command `ps aux`. This command provides a list of all the processes that are currently running on your computer. You'll notice that the list doesn't fit on the screen.
3. To make sure you can see the complete result page by page, use `ps aux | less`. The output of `ps` is now sent to `less`, which outputs it so that you can browse it page by page.

## Redirecting Output to a File

1. From a console window, use the command `ps aux`. You'll see the output of the command on the current console.
2. Now use `ps aux > ~/psoutput.txt`. You don't see the actual output of the command, because it is written to a file that is created in your home directory, which is designated by the `~` sign.
3. To show the contents of the file, use the command `less ~/psoutput.txt`.

# **System Administration Tasks**

# Performing Job Management Tasks

- While some commands take only a few seconds or less to finish, other commands may take much longer.
- By putting an **&** sign at the end of a command, you start it as a background job.
- When starting a command this way, the **shell provides a job number (between square brackets) and a unique process identification number (the *PID*)**.
- You can then use these numbers to manage your background jobs.

**TABLE 3.1** Managing foreground and background jobs

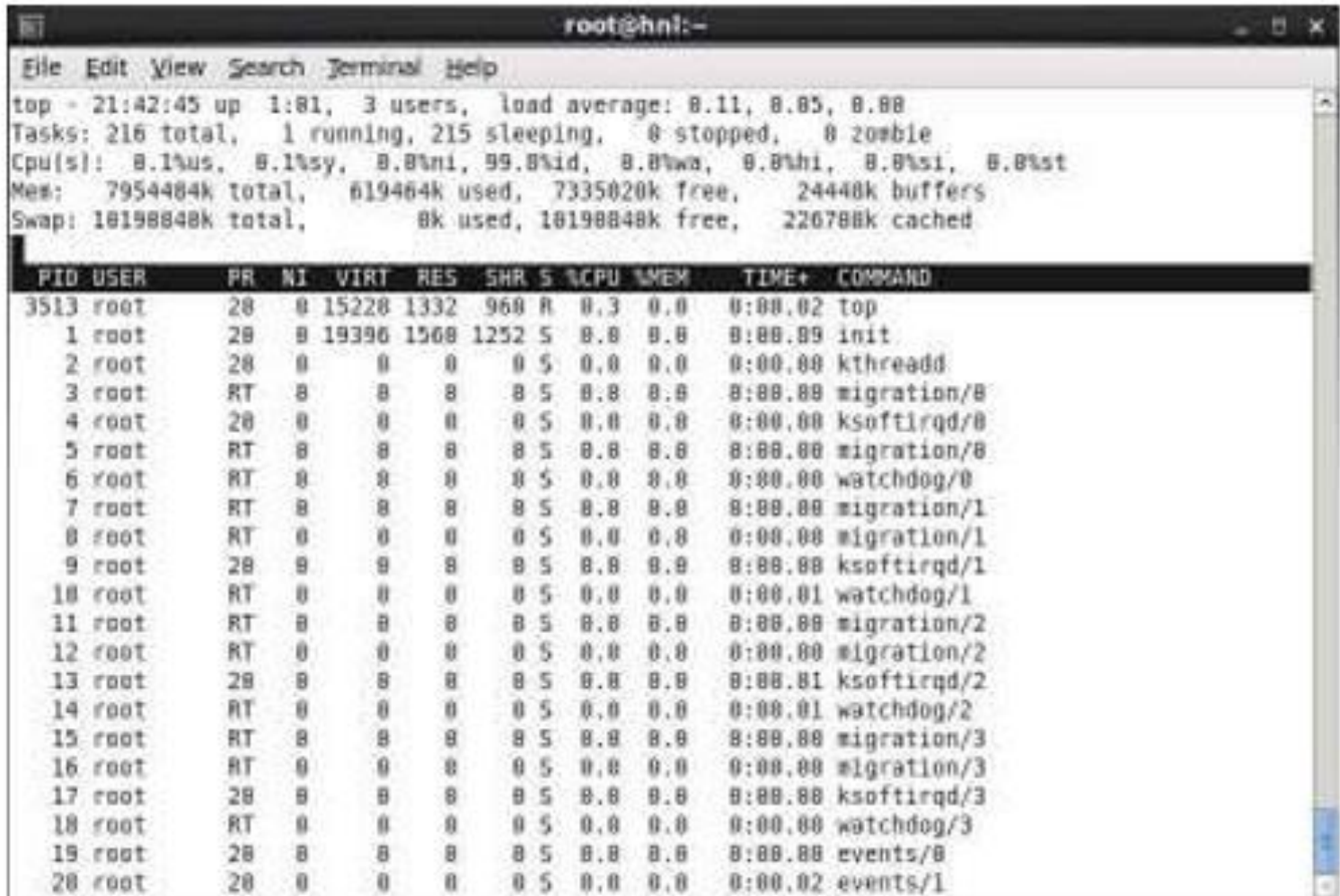
Command	Use
Ctrl+Z	Use this to pause a job. Once paused, you can put it in the foreground or in the background.
fg	Use this to start a paused job as a foreground job.
bg	Use this to start a paused job as a background job.
jobs	Use this to show a list of all current jobs.

# Sending Signals to Processes with the kill Command

- Three signals are available at all times: SIGHUP (1), SIGKILL (9), and SIGTERM (15).
- Each of these signals can be referred to by the name of the signal or by the number when managing processes. You can, for instance, use either **kill -9 123** or **kill -SIGKILL 123** to send the SIGKILL signal to the process with PID 123.
- If, as an administrator, you request closure of a program using the **SIGTERM** signal, the process in question can still close all open files and stop using its resources.

# Using top to Show Current System Activity

**FIGURE 3.3** Showing current system activity with top



```
root@hnl:~  
File Edit View Search Terminal Help  
top - 21:42:45 up 1:01, 3 users, load average: 0.11, 0.05, 0.00  
Tasks: 216 total, 1 running, 215 sleeping, 0 stopped, 0 zombie  
Cpu(s): 0.1%us, 0.1%sy, 0.0%ni, 99.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st  
Mem: 7954404k total, 619464k used, 7335020k free, 24448k buffers  
Swap: 10190048k total, 8k used, 10190048k free, 226700k cached  


| PID  | USER | PR | NI | VIRT  | RES  | SHR  | S | %CPU | %MEM | TIME+   | COMMAND     |
|------|------|----|----|-------|------|------|---|------|------|---------|-------------|
| 3513 | root | 20 | 0  | 15228 | 1332 | 960  | R | 0.3  | 0.0  | 0:00.02 | top         |
| 1    | root | 20 | 0  | 19396 | 1560 | 1252 | S | 0.0  | 0.0  | 0:00.09 | init        |
| 2    | root | 20 | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | kthreadd    |
| 3    | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/0 |
| 4    | root | 20 | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | ksoftirqd/0 |
| 5    | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/0 |
| 6    | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | watchdog/0  |
| 7    | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/1 |
| 8    | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/1 |
| 9    | root | 20 | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | ksoftirqd/1 |
| 10   | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.01 | watchdog/1  |
| 11   | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/2 |
| 12   | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/2 |
| 13   | root | 20 | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.01 | ksoftirqd/2 |
| 14   | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.01 | watchdog/2  |
| 15   | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/3 |
| 16   | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | migration/3 |
| 17   | root | 20 | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | ksoftirqd/3 |
| 18   | root | RT | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | watchdog/3  |
| 19   | root | 20 | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | events/0    |
| 20   | root | 20 | 0  | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.02 | events/1    |


```



- In the upper five lines of the top interface, you can see information about the current system activity.
- The lower part of the top window shows a list of the most active processes at the moment. This window is refreshed every five seconds.
- If you notice that a process is very busy, you can press the k key from within the top interface to terminate that process. The top program will first ask for the PID of the process to which you want to send a signal (PID to kill). After you enter this, it will ask which signal you want to send to that PID, and then it will immediately operate on the requested PID.

In the second line of the top window, you'll see how many tasks your server is currently handling and what each of these tasks is doing. In this line, you may find four status indications.

running	The number of active processes in the last polling loop.
sleeping	The number of processes currently loaded in memory, which haven't issued any activity in the last polling loop.
stopped	The number of processes that have been sent a stop signal but haven't yet freed all of the resources they were using.
zombie	The number of processes that are in a zombie state. This is an unmanageable process state because the parent of the zombie process has disappeared and the child still exists but cannot no longer be managed because the parent is needed to manage that process.

# Managing Process Niceness

- When using the nice command, you can **adjust the process niceness** from **-20**, which is good for the **most favorable scheduling**, to **19** for the **least favorable scheduling**.
- By default, all processes are started with a niceness of 0.
- Aside from specifying which niceness setting to use when starting a process, you can also use the **renice** command to adjust the niceness of **a command that has already started**.
- By default, renice works on the PID of the process whose priority you want to adjust. Thus, you have to find this PID before using renice.

# Scheduling Jobs

- Think, for example, of a backup job that you want to execute automatically every night. To start jobs automatically, you can use cron.
- cron consists of two parts. First there is the cron daemon, a process that starts automatically when your server boots.
- The second part is the cron configuration. This is a set of different configuration files that tell cron what to do.
- The cron daemon checks its configuration every minute to see whether there are any new tasks that should be executed.

# Mounting Devices

- As an administrator, you'll occasionally need to make storage devices like USB flash drives, hard drives available.
- To do this, you need to connect the device to a directory in the root file system. **This process is known as mounting the device.**
- devices are mounted automatically on graphical desktop.
- **To mount a storage device, you first need to find out two things:** what is the name of the device you want to mount, and on which directory do you want to mount it?
- Normally, the **primary hard drive in your server is known as /dev/sda.** However, if your server is connected to a SAN, you might have many additional sd devices

# Working with Links

- It is very useful to be able to access a single file from different locations.
- In a Linux file system, you can use links for this purpose. A link appears to be a regular file, but it's more like a pointer that exists in one location to show you how to get to another location.
- In Linux, there are **two** different types of **links**. A **symbolic link** is the most flexible link type you can use. It **points to any other file and any other directory**, no matter where it is. A **hard link** can be used only to point to a file that exists on the same device.
- With **symbolic links**, there is a difference between the original file and the link. If you remove the original file, the symbolic link won't work anymore and thus is invalid.

# Creating Backups

- you might want to make a backup of important files on your computer. The **tar** command is the most common way of creating and extracting backups on Linux

## Archiving and Extracting with tar

In this exercise, you'll learn how to archive the contents of the `/etc` directory into a tar file. Next you'll check the contents of the archive, and as the last step, you'll extract the archive into the `/tmp` directory.

1. Open a terminal, and use the following command to write an archive of the `/etc` directory to `/tmp/etc.tar`: `tar zxvf /tmp/etc.tar /etc`.
2. After a short while, you'll have a tar archive in the `/tmp` directory.
3. Use the command `file /tmp/etc.tar` to verify that it is indeed a tar archive.
4. Now show the contents of the archive using `tar tvf /tmp/etc.tar`.
5. Extract the archive in the `/tmp` directory using `tar xvf /tmp/etc.tar`. Once finished, the extracted archive is created in the `/tmp` directory, which means you'll find the directory `/tmp/etc`. From there, you can copy the files to any location you choose.

# Setting Up Logrotate

- On a very busy server, you may find that **entries get added to your log files really fast.**
- your **server may quickly become filled with log messages**, leaving little space for regular files.
- two solutions to this problem:
  - First, the directory `/var/log` should be on a **dedicated partition** or logical volume.
  - Another solution is using **logrotate**.



- By default, the logrotate command runs as a cron job once a day from [/etc/cron.daily](#).
- Rotating a log file basically means that the **old log file is closed and a new log file is opened**.
- logrotate keeps a certain number of the old logged files, often stored as compressed files on disk.
- When the **maximum amount** of old log files is reached, **logrotate** **removes them automatically**.

# Understanding RPM

- In the early days of Linux, the “tar ball” was the default method for installing software. A tar ball is an archive that contains files that need to be installed.
- Working with tar balls was inconvenient for several reasons.
  - There was no standardization.
  - When using tar balls, there was no way to track what was installed.
  - Updating and de-installing tar balls was difficult to do.
- The ability to trace software was needed to overcome the disadvantages of tar balls. The Red Hat Package Manager (RPM) is one of the standards designed to fulfill this need.

- An RPM is basically **an archive file**. It is created with the **cpio** command.
- With RPM, there is also **metadata** describing what is in the package and where those different files should be installed.
- Another benefit of using RPM is that its database is created in the **/var/lib/rpm** directory.
- This database **keeps track** of the exact version of files that are installed on the computer.
- Thus, for an administrator, it is **possible to query individual RPM files** to see their contents.

# Creating Your Own Repositories

- If you have a Red Hat server installed that **doesn't have access to the official RHN repositories**, you'll need to set up your own repositories.
- This procedure is also useful if you want to copy all of your RPMs to a directory and use that directory as a repository.
- Exercise 4.1

# Managing Repositories

- you learned how to turn a directory that contains RPMs into a repository.
- However, just marking a directory as a repository isn't enough.
- To [use your newly created repository](#), you'll have to tell your server where it can find it.
- To do this, you need to create a repository file in the directory `/etc/yum.repos.d`. You'll probably already have some repository files in this directory.
- Exercise 4.2

# Installing Software with Yum

- After configuring the repositories, you can **install, query, update, and remove software** with the meta package handler yum.
  - Searching Packages with Yum
  - Installing and Updating Packages
  - Removing Packages

# Installing and Updating Packages

- Once you've found the package you were seeking, you can install it using `yum install`.
- For instance, if you want to install the network analysis tool nmap, you'd use `yum install nmap` to install the tool.
- Yum will then `check the repositories` to find out where it can `find the most recent version` of the program you're seeking, and after finding it, yum shows you what it wants to install.
- If there are `no dependencies`, it will show just one package. However, if there are `dependencies`, it displays a list of all the packages it needs to install in order to give you what you want. Next, `type Y to confirm` that you really want to install what yum has proposed, and the software will be installed.

- Two useful options when working with yum install
  - first option, -y: Use `yum install -y` to proceed immediately, without any additional prompts for confirmation.
  - Another useful yum option is --nogpgcheck: For instance, use `yum install -y --nogpgcheck xinetd` if you want to install the xinetd package without performing a GPG check and without having to confirm the installation.



# Removing Packages

- Just use `yum remove` followed by the name of the package you want to uninstall.
- For instance, to remove the package `nmap`, use `yum remove nmap`.
- The `yum remove` command will first provide an overview of what exactly it intends to do.
- In this overview, it will display the name of the package it intends to remove and all packages that depend on this package.
- It is very important that you read carefully what `yum` intends to do. If the package you want to remove has many dependencies, by default `yum` will remove these dependencies as well

# Querying Software

- There are many **ways to query** software packages.
- You can query **packages that are currently installed** on your system, and it's also possible to install **package files that haven't yet been installed**.
- To query an installed package, you can use one of the **rpm -q** options.
- To get information about a package that hasn't yet been installed, you need to add the **-p option**.
- Example :
  - **rpm -ql samba-common** command, if this package is installed
  - In case it hasn't yet been installed, you need to use **rpm -qpl samba-common-[version-number].rpm**

- `rpm -qa` : this command generates a list of all RPM packages that are installed on your server and thus provides a useful means for finding out whether some software has been installed.
- if you want to **check whether the media-player package is installed**, you can use `rpm -qa | grep mediaplayer`.
- A useful modification to `rpm -qa` is the **-V option**, which shows you if a **package has been modified from its original version**.
- `rpm -qVa` : Every file that is shown in the output of this command has been modified since it was originally installed