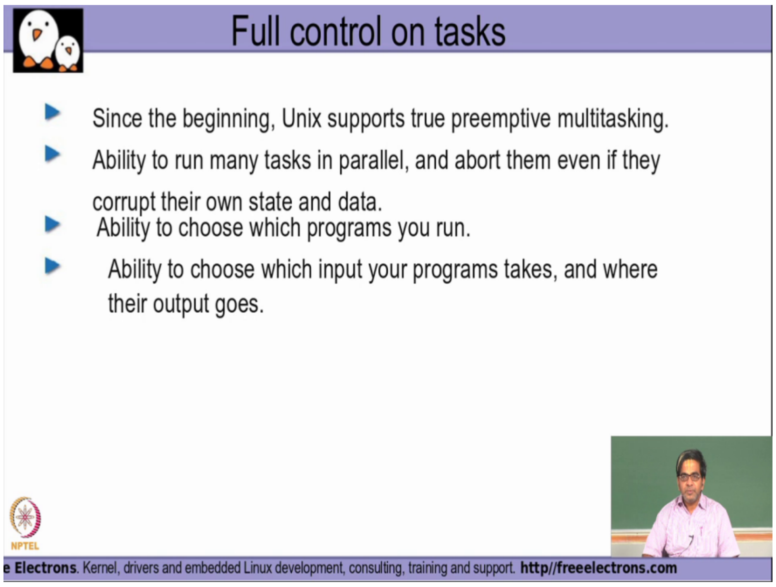
Information Security
Shri Vasan V S, Principal Consultant
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Module 19
LINUX Task Control

So this module we will actually be talking about the different kinds of control mechanism that we have for starting a task and managing a task once it is actually started in the command line.

(Refer Slide Time: 00:26)



Full control on tasks

- Since the beginning, Unix supports true preemptive multitasking.
- Ability to run many tasks in parallel, and abort them even if they corrupt their own state and data.
- Ability to choose which programs you run.
- Ability to choose which input your programs takes, and where their output goes.

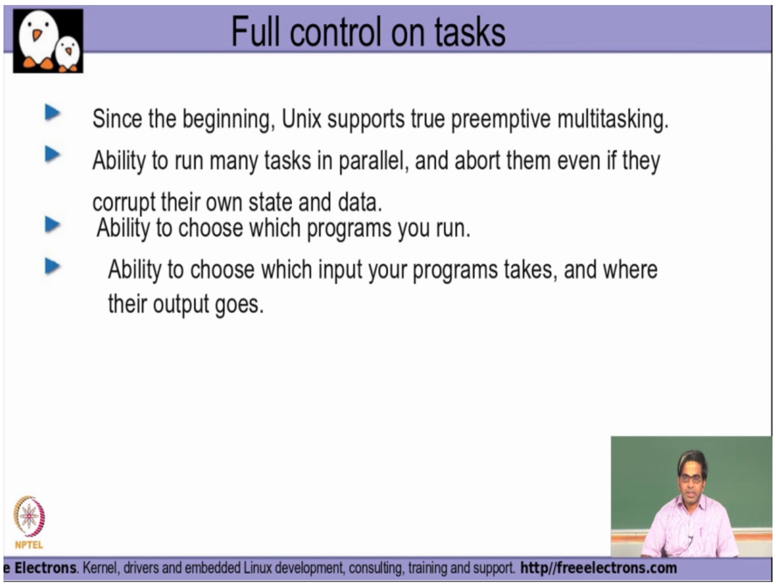
e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So as we were discussing earlier LINUX operating system is sort of supporting through pre-emptive multitasking. So now what is exactly mean by True pre-emptive multitasking will just do a quick recap .

So multitasking is a mechanism by which the system will be able to run in task independently and same time multiple task could be run and then pre-emptive essentially means that the system is not going to be allowing 1 task to be waiting for a long period of time just because another task has been running for a certain amount of time.

So the system is going to preamp the current running task and give the processor to the another task that is actually waiting for for to be running and thereby the Unix OS is actually treated as a very very true a pre-emptive multi tasking system. So its a mechanism by which I will be able to run multiple multiple tasks in parallel at the same time.

(Refer Slide Time: 01:31)



Full control on tasks

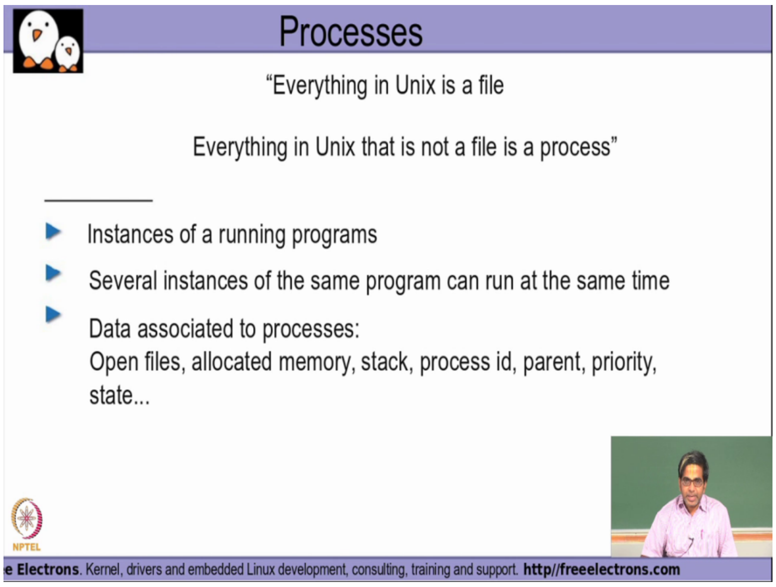
- Since the beginning, Unix supports true preemptive multitasking.
- Ability to run many tasks in parallel, and abort them even if they corrupt their own state and data.
- Ability to choose which programs you run.
- Ability to choose which input your programs takes, and where their output goes.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

And also about them if they basically try to correct any any other portion of the memory for which it is actually doesn't have any rights right? so there could be some processes that inadvertently tries to Access the Memory area of another process for which it does not have a right, so all those kind of processes will be immediately killed by the OS not making it run subsequently after that.

It also has a mechanism by which it will be able to choose which program should we needed to run at any point in time and also what input the programs might take and where the output basically goes to. So this is basically what we were seeing as the input redirection and output redirection as well as other re direction in our earlier module.

(Refer Slide Time: 02:23)



Processes

"Everything in Unix is a file

Everything in Unix that is not a file is a process"

- Instances of a running programs
- Several instances of the same program can run at the same time
- Data associated to processes:
  Open files, allocated memory, stack, process id, parent, priority, state...

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

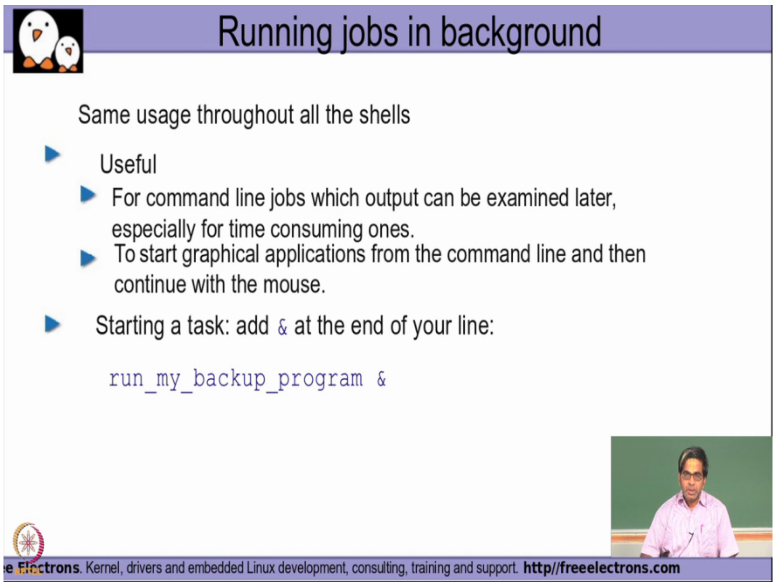
Now what exactly process in Unix as , so as we were discussing in our earlier module we said that everything in a Unix is a file let's put in a different way in everything in Unix is not a file is a process right? So what is mean by this is that a process is something which is actually alive and running on Unix whereas a file is something that is statically present in some part of my system without having a so-called life in it right?

So its going to be sort of resident in some corner of my let us say storage mechanism whether it be my pendrive or my hard disc or whatever it is. Whereas if that file is for example is an executable file and I go start that file by either typing into the command line or by double clicking on it in my bro in my Explorer window the moment the file is actually read from my hard disc location and then gets loaded into my memory that file basically magically becomes a running process right?

So Unix basically now creates a process copies the content of the executive instructions in the file makes it as a Process makes it run and then the beauty here is that I could actually have several instances the same program running at the same time.

So for example everybody wants to run the program called LS on a system. So let's say there are five users logged in the system and at the milli second granularity time frame all the five users logged in the system currently wants to run the same LS command all of them will be able to execute the same command same time, right? So there in the system basically supports multiple instances the same program getting executed the same time in a very very seamless manner. So there are different types of data that is typically associated the process which we would have actually seen very briefly when we talked about the different parts of the operating system initially.

(Refer Slide Time: 04:30)



Running jobs in background

Same usage throughout all the shells

- Useful
  - For command line jobs which output can be examined later, especially for time consuming ones.
  - To start graphical applications from the command line and then continue with the mouse.
- Starting a task: add & at the end of your line:

```
run_my_backup_program &
```

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So you have what are the open files are maintained by the Process what is the amount of memory that has been actually allocated stack the process ID being the unique ID identified that is identified that through which the process is actually identified in system so what is a parent process what is a Priority the process what is the current state the process like whether it is sleeping whether it is running or whether it is waiting for IO to complete or whether it is in a Zombie state all those kind of data are associated with each and every process that is actually running in a system.

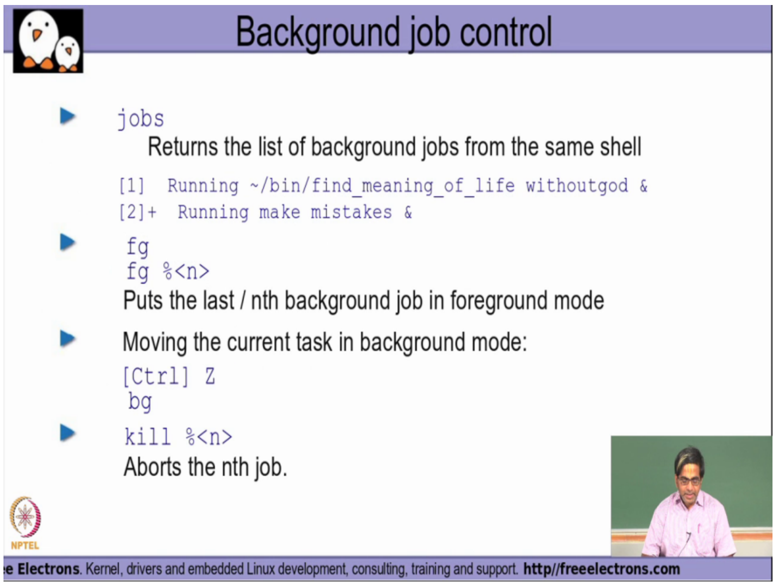
So because of the fact that Unix is basically supporting multitasking one very important features that is actually being made available on Unix is to make a particular task run in the background. Now what is mean by running in the background is that by default when you type a command in the command line and press the enter key this command is no going to be read and loaded in the memory and that is going to become a process and starts running.

So by default we tell that this command is actually running in the foreground because they unless and until the command is completed so the command will not be complete without without which without the command getting completed you will not be able to give any other command or job for the shell to run that is typically what we mean when we say the command is actually running in the phorground now for the command to run in the background the main reason for giving this kind of functionality is that we will understand that there will be certain commands that will actually take a very very long time to complete.

Now I will be requiring certain other jobs to be done at the same time when this command is being run in the background so that I don't have to wait for a long period of time for submitting the next command to the system so that is basically the reason why the OS is actually supporting a mechanism by which a job could be given to be running in the background and thereby immediately getting the prompt back for the user to give the next command to the shell.

So if I have this kind of requirement how do I typically get it done I use the ampersand character what we call as the and operator also in the programming parlance I use the character at the end of the line to denote to my shell that this command should be run as part of a background task and not as part of the foreground task so whenever I give this kind of ampersand character the shell will retreat that this command should be run in the background start this process make it as a background job and then immediately come back to the shell from so that the user can basically type the next command or the next tool that he wants to run as part of using the shell so that is basically The main advantage in the reason behind why the the Unix operating system gives you a mechanism to run a task in the background.

(Refer Slide Time: 07:49)



Background job control

- `jobs`
  Returns the list of background jobs from the same shell

```
[1]  Running ~/bin/find_meaning_of_life withoutgod &
[2]+  Running make mistakes &
```

- `fg`
  `fg %<n>`
  Puts the last / nth background job in foreground mode
- Moving the current task in background mode:
  `[Ctrl] Z`
  `bg`
- `kill %<n>`
  Aborts the nth job.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

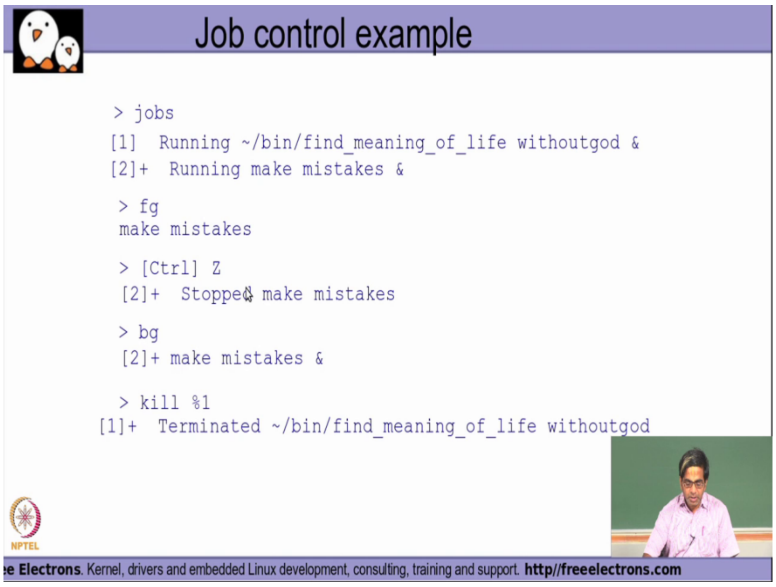
So when I’m running multiple task in the background I would need to know at any point in time what are the different tasks that is running. So especially if I have basically given different tasks and I want to know what is the task that has been given till date and what are the status of each of that and that is the reason why we have the command called jobs. So if I say jobs it will basically return me back and display the list of the current background jobs it is running.

So out of the listed task if I want to basically bring any of those task to the foreground I can use the fg command where in I say FG followed by the percentage and the number. So if I want to bring down the second command I say fg percent 2 and then the number will be the number that s actually getting matched with what is printed as part of the output of my jobs command. So if I want to bring the second command out as the foreground process I say fg percent 2 right?

And if I don't give this percent it basically tries to put the last command that has been given in the background into the foreground. Now after the task has sufficiently run in the foreground and I want to shift it back into the background I say control Z; control Z essentially is a temporary suspend character the suspend key that will give in the Shell to the currently running process after which I can type the command called bg; bg means to put the currently suspended task in the back end right in the background .

So this is the equivalent to running the command with the ampersand character at the time of starting the command Excel right? Now another command that we need to understand as part of the job control is a command called kill which is typically used to kill the command that is associated with that particular background job. So if I say kill followed by percent 1 it will basically try to kill this particular task that has actually been listed as a first process as part of the job output .

(Refer Slide Time: 10:07)



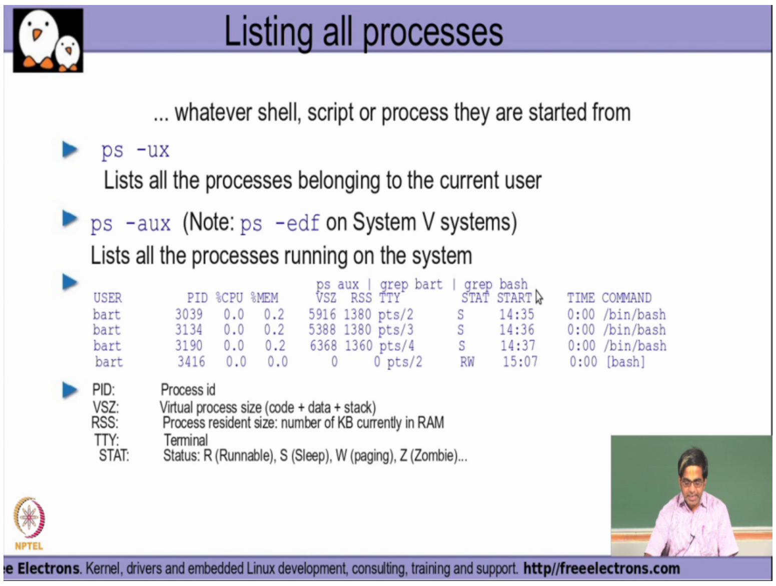
Job control example

```
> jobs
[1]  Running ~/bin/find_meaning_of_life withoutgod &
[2]+  Running make mistakes &

> fg
make mistakes

> [Ctrl] Z
[2]+  Stopped make mistakes

> bg
[2]+ make mistakes &

> kill %1
[1]+  Terminated ~/bin/find_meaning_of_life withoutgod
```

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So that will basically about the the job that has been passed here right? So again an example of how you could typically make use of this. So if I say kill percent 1 it reports back terminated and then tells me whatever was a job that was actually given for that particular background task.

(Refer Slide Time: 10:25)



Listing all processes

... whatever shell, script or process they are started from

- `ps -ux`
  Lists all the processes belonging to the current user
- `ps -aux` (Note: `ps -edf` on System V systems)
  Lists all the processes running on the system
-
```
                                 ps aux | grep bart | grep bash
USER          PID %CPU %MEM      VSZ  RSS TTY          STAT START    TIME COMMAND
bart         3039  0.0  0.2     5916 1380 pts/2        S    14:35    0:00 /bin/bash
bart         3134  0.0  0.2     5388 1380 pts/3        S    14:36    0:00 /bin/bash
bart         3190  0.0  0.2     6368 1360 pts/4        S    14:37    0:00 /bin/bash
bart         3416  0.0  0.0        0    0 pts/2        RW   15:07    0:00 [bash]
```
- PID: Process id
  VSZ: Virtual process size (code + data + stack)
  RSS: Process resident size: number of KB currently in RAM
  TTY: Terminal
  STAT: Status: R (Runnable), S (Sleep), W (paging), Z (Zombie)...

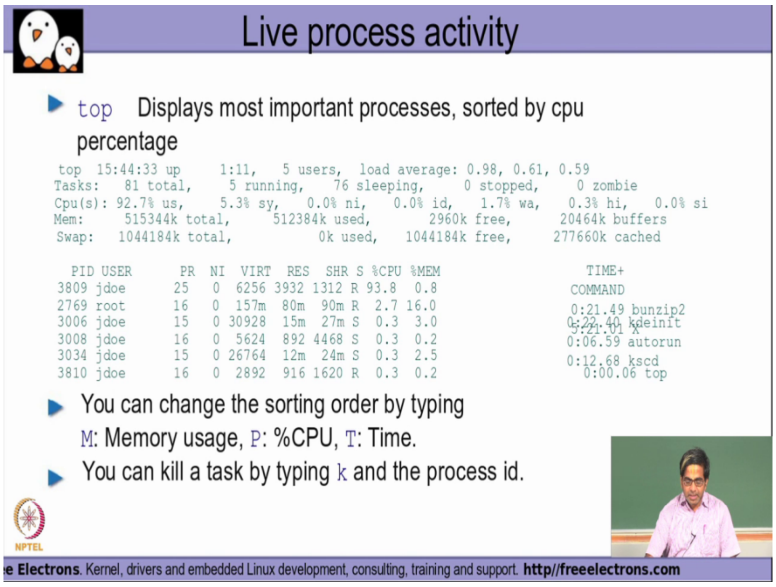
e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So if I want to find out the different processes that are right now running on the system either by myself or on the entire system I could actually use the minus aux option to the PS command on minus edf option on some system based systems where in it will bring me the complete output which will contain on a per process basis what are the what is the user id which has actually started this particular process what is the PID that is a process ID how much of CPU has been used?

How much memory has been used? What is the virtual process size? What is a resident process size? what is a terminal on which it has been started? What is the current status? Whether it is running whether it is getting sleepy? whether it is currently sleeping waiting for let us say some I/O to complete or whether it is getting page or whether it is a zombie process, right?

So when was it started how much time it has actually run and what is the corresponding command. So for every process in the currently running system with these options to the ps command we get all these details display which gives sense of where each command is currently and what kind of resources each command is actually occupying.

(Refer Slide Time: 11:46)



Live process activity

- `top` Displays most important processes, sorted by cpu percentage

```
 top  15:44:33 up    1:11,   5 users,  load average: 0.98, 0.61, 0.59
Tasks:   81 total,    5 running,   76 sleeping,     0 stopped,    0 zombie
Cpu(s): 92.7% us,    5.3% sy,   0.0% ni,   0.0% id,   1.7% wa,   0.3% hi,   0.0% si
Mem:     515344k total,     512384k used,       2960k free,     20464k buffers
 Swap:   1044184k total,          0k used,   1044184k free,    277660k cached

  PID USER       PR  NI  VIRT  RES  SHR S %CPU %MEM                    TIME+
 3809 jdoe       25   0  6256 3932 1312 R 93.8  0.8                   COMMAND
 2769 root       16   0  157m  80m  90m R  2.7 16.0                   0:21.49 bunzip2
 3006 jdoe       15   0 30928  15m  27m S  0.3  3.0                  0:22.40 kdeinit
 3008 jdoe       16   0  5624  892 4468 S  0.3  0.2                   5:21.01 X
 3034 jdoe       15   0 26764  12m  24m S  0.3  2.5                  0:06.59 autorun
 3810 jdoe       16   0  2892  916 1620 R  0.3  0.2                  0:12.68 kscd
                                                                        0:00.06 top
```

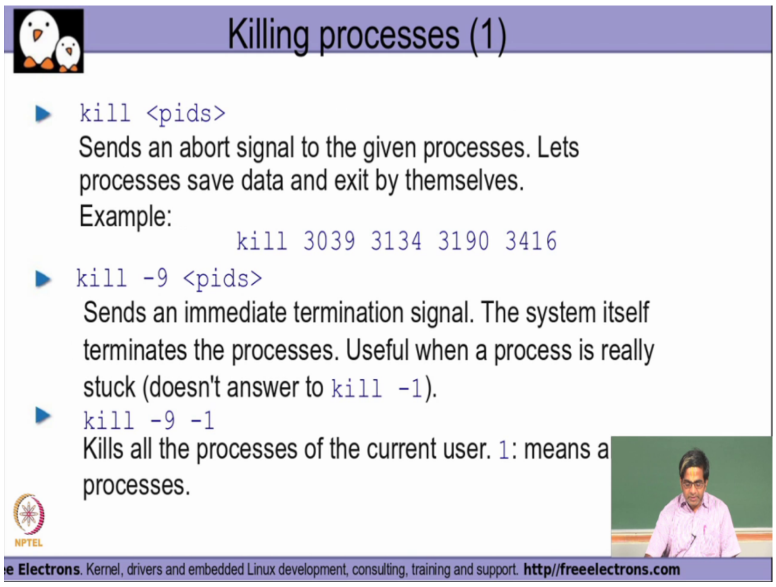
- You can change the sorting order by typing `M`: Memory usage, `P`: %CPU, `T`: Time.
- You can kill a task by typing `k` and the process id.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. **http//freeelectrons.com**

So talk is another command that will give me the complete process activity on a system the most important feature of the top command is that for every few seconds typically like around 5 seconds the output will be getting automatically refresh so if I really want to find out what are the type of processes that are actually consuming the maximum amount of CPU or the maximum amount of memory at any point in time in my system then this command is something which is extremely useful for system administrator wherein he tries to print out the details in a very very dynamic manner and gets continuously refresh every few seconds or so;

so that with this kind of information the system administrator typically will be able to find out what are the kind of processes that are sort of consuming the maximum amount of resources, right? So if I want to basically kill a particular task which is actually getting listed in the top output I could just press the key k and then followed by the process ID and then it will automatically try to kill that particular process assuming that this particular person has permission to kill the process.

(Refer Slide Time: 13:00)



Killing processes (1)

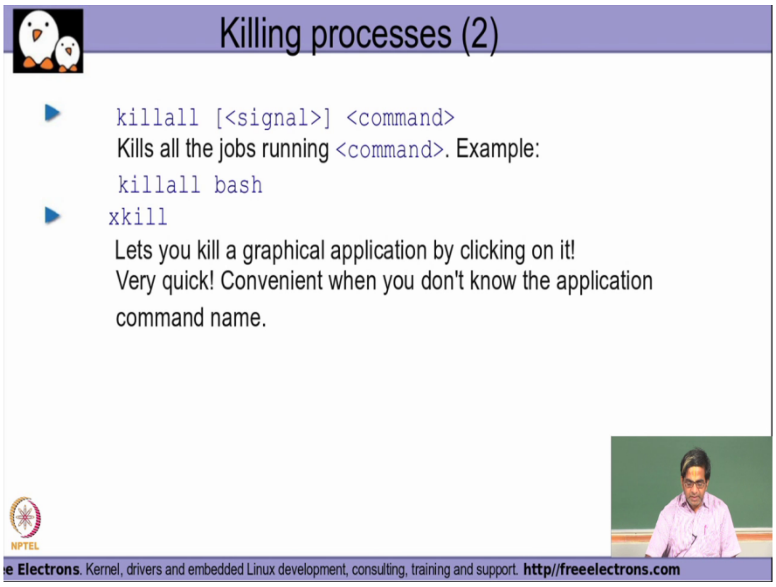
- `kill <pids>`
  Sends an abort signal to the given processes. Lets processes save data and exit by themselves.
  Example: `kill 3039 3134 3190 3416`
- `kill -9 <pids>`
  Sends an immediate termination signal. The system itself terminates the processes. Useful when a process is really stuck (doesn't answer to `kill -1`).
- `kill -9 -1`
  Kills all the processes of the current user. `1`: means a
  processes.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So otherwise I could also get the process kill by using different commands I could use kill followed by the PIDs to the kill command followed by the PIDs will try to send a kill signal to each of those PIDs. So when I say kill minus 9 followed by PIDs it will send an immediate termination signal what we call as a Sig term to each of the the the process IDs that is mentioned here and assuming that the user who is running the kill command has permission to kill that particular process ID or AP is a Super user so if either of this condition is satisfied then that particular set of PIDs that is mentioned here one or more PIDs that is mentioned here will be killed successfully by the system. So if I say kill minus 9 minus 1 it will kill all the processes the current user that is being currently run.

So if a user called user 1 as actually logged in and they have started like 5 different commands of 5 different applications and after that if he has basically running the command for kill minus 9 minus 1 all those process applications will be getting killed and then he will come back to the state whatever he was in before even if he had actually logged into the system. Because even the shell command that was actually available to him as part of a successful login that also will be getting killed if this particular come and kill is run with minus 9 minus 1 the shell process also is the process started by that particular user because of which the cell process also will be getting killed with this particular command.

(Refer Slide Time: 14:46)



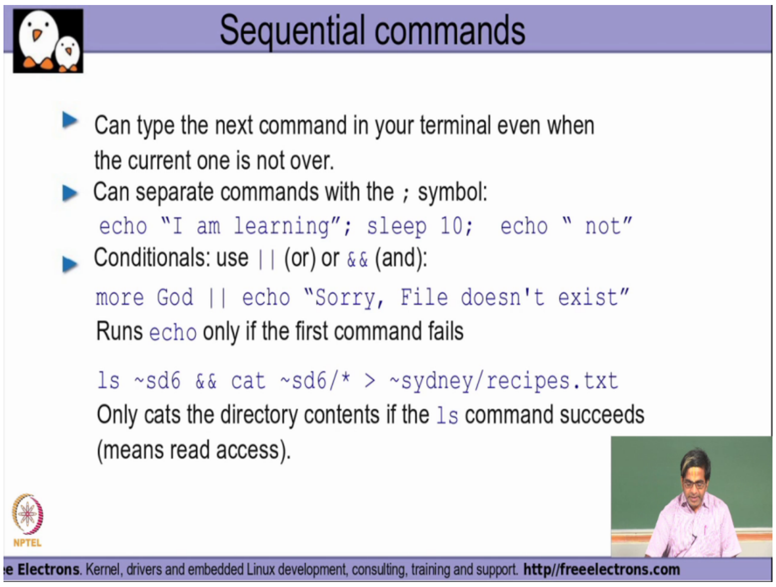
Killing processes (2)

- `killall [<signal>] <command>`
  Kills all the jobs running `<command>`. Example:
  `killall bash`
- `xkill`
  Lets you kill a graphical application by clicking on it!
  Very quick! Convenient when you don't know the application command name.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

Alternatively we can also kill all or xkill to kill all the processes which are running this particular command again I should have permission as a Superuser typically or as a normal user if all the commands at a running as processes right now have been started by me then I will be able to run the kill all come and successfully on all those processes right? xscale is actually used to kill any kind of a graphical application that you might be running.

(Refer Slide Time: 15:19)



Sequential commands

- Can type the next command in your terminal even when the current one is not over.
- Can separate commands with the `;` symbol:
  `echo "I am learning"; sleep 10;  echo " not"`
- Conditionals: use `||` (or) or `&&` (and):
  `more God || echo "Sorry, File doesn't exist"`
  Runs `echo` only if the first command fails

  `ls ~sd6 && cat ~sd6/* > ~sydney/recipes.txt`
  Only cats the directory contents if the `ls` command succeeds (means read access).

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So the next way of the way that I typically used commands as part of the job control is running them in a sequential manner. So what we mean by sequential manner is if I want to run multiple commands together in a single command line I could use semicolon as as as a separator and then run them so here it will first run the command echo that is given here in this example and since there is a semi colon it is right to run the second command then there is a semi colon it will run the third command.

So in this case each command which has been given in the command line delimited by semicolon will be run one after the other by the shell and in that way it is also a sort of a job control where in they are Run one after the other sequentially even though there are being multiple commands given in a single command line. Now another way of running the sequential commands is using what is called as a logical or the logical and operator when each of these commands run and when they do not run is actually dependent on what the what is the status of the the previous command execution.

So if I basically say that I want to run this first command more God. So I will try to display the contents of this file call God and if that particular command is not successful when will it not be successful when this particular file does not exist right? what it will actually do is it will now encounter the logical or, so as far as the truth table of or is concerned the overall value result value of the entire expression is true if either one is actually true right?

So because of the fact that the first one that has been given as part of this logical or expression is false in a scenario where this particular file is not existing the shell will actually try to go and execute the second command that is actually been given here, right? And what is a command that is actually been given here it is an echo command where in this particular message is going to be getting displayed right?

On the other hand if the first one was successful because of the fact that one of them has been successful out of the the first Command or the second command the shell will not go and execute the second command because for the logical truth table of an or condition is one of them is true then it is actually sufficient, right? So in this particulars scenario the second command is actually going to be getting executed only if the first one is actually failing.

On the other hand when we actually say you want use and operator, the and operator will be done only if the previous one has been successful because like we were saying the truth table of or condition for the and condition the truth table essentially means that the whole value is true only if the resultant Sub expressions are true So in this particular case if you see we have said that they want to list the contents of the home directory of the SD6 user and for example if this SD6 user is not existing or the home directory is not existing there is no point in trying list the contents of this sub directories or files of this home directory, right? So that is the reason why the logical and operator is used here.

So if this was successful then I will have the second command getting executed also and the output of this command is also redirected to another file. So because of the fact that the logical truth table value of an and operation will be 1 only both of them are 1 The if only if the first condition first command is actually successful the shell will try to go and execute the second part of the the logical conditional operator expression and if this was not successful for any reason maybe that SD6 was not a valid user because we are trying to with the users it is tilde characters you are trying to refer to the home directory of the SD6 user.

So if SD6 user is not existing nonexistent user there will be no home directory of SD6 user or if the SD6 user is a real user but there's been no home directory configure this particular user in either case this condition will fail if this condition fails then there is no point in actually running it and therefore we are using and operator the and operator here and because for the and operation both of them as successful and in this particular case it has been a failure the shell will not try to run the second part of the the second expression that has been given here also.

On the other hand if this has been a successful where in the SD6 is valid user and we also have a home directory for the sd6 user then that sell will actually try to run the second part of the expression also because and operation is given and this and operation will be running when we have the first part of the expression to be a true resultant value so in this way we can actually use conditional operator like or or and to make the commands executed in a sequential manner depending on what has been the output of the previous command the next command can either be run or not be run by using the appropriate logical conditional operator Thank you!