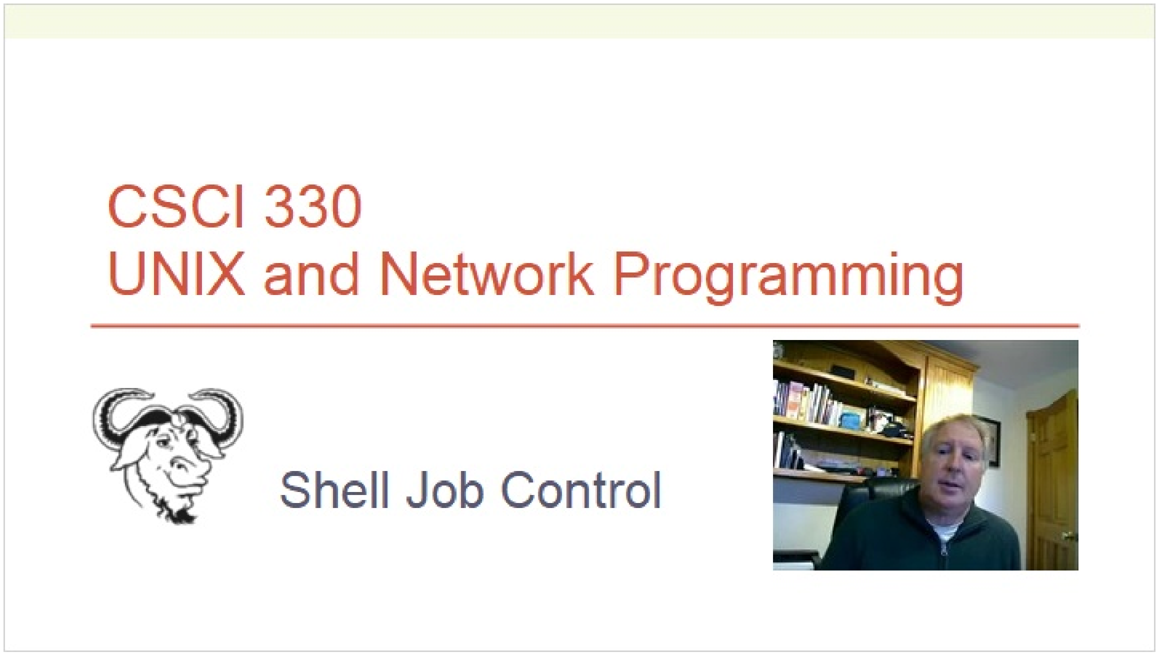
Shell Job Control

1. Shell Job Control

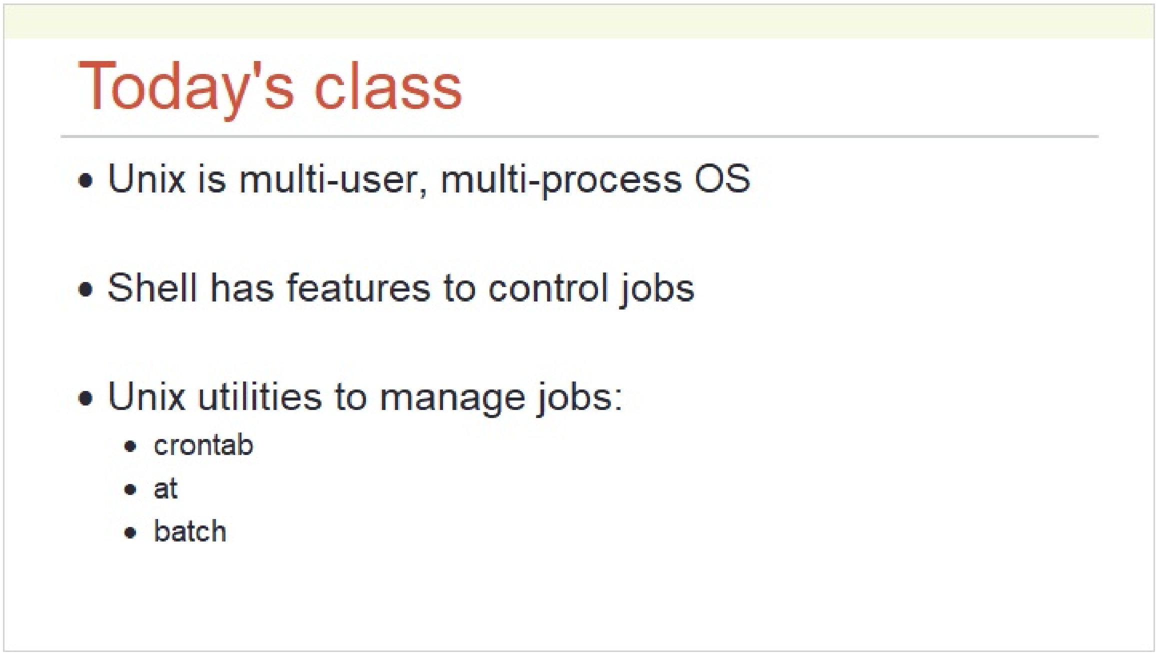
1.1 Introduction



CSCI 330
UNIX and Network Programming

Shell Job Control

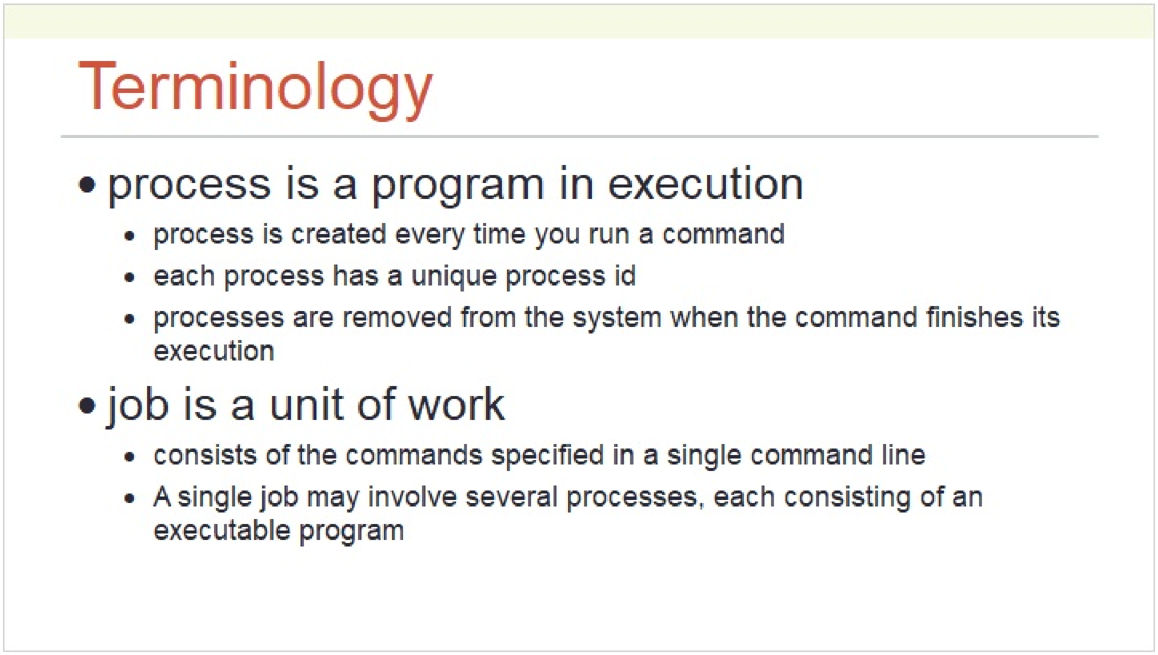
1.2 Today's class



Today's class

- Unix is multi-user, multi-process OS
- Shell has features to control jobs
- Unix utilities to manage jobs:
  - crontab
  - at
  - batch

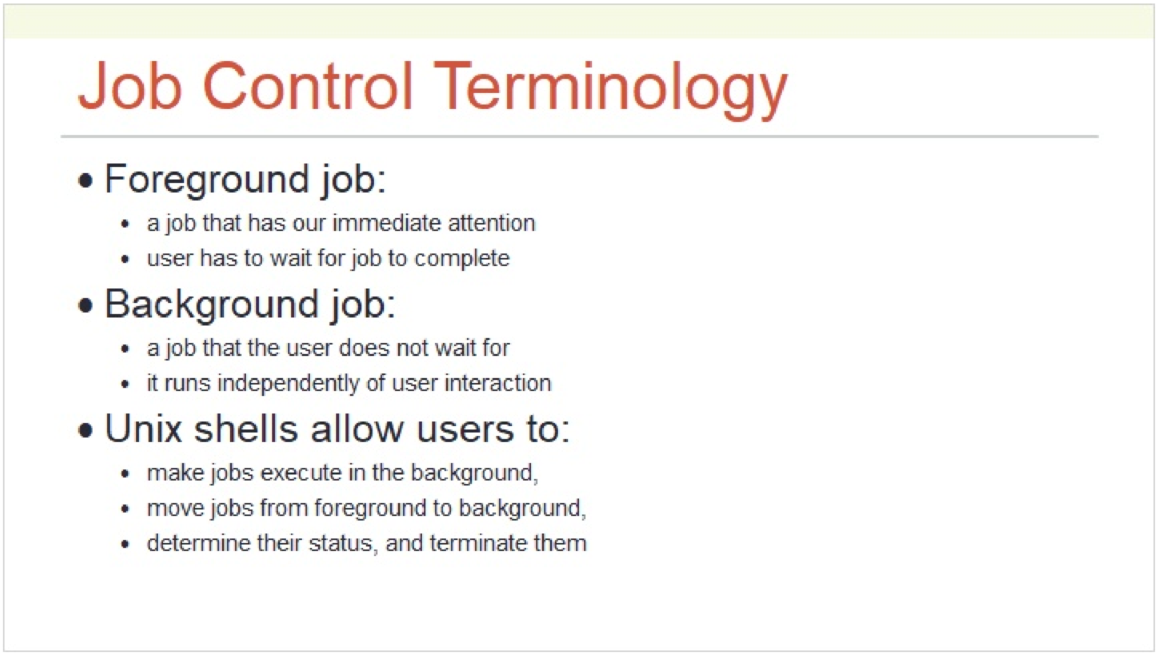
1.3 Terminology



Terminology

- process is a program in execution
  - process is created every time you run a command
  - each process has a unique process id
  - processes are removed from the system when the command finishes its execution
- job is a unit of work
  - consists of the commands specified in a single command line
  - A single job may involve several processes, each consisting of an executable program

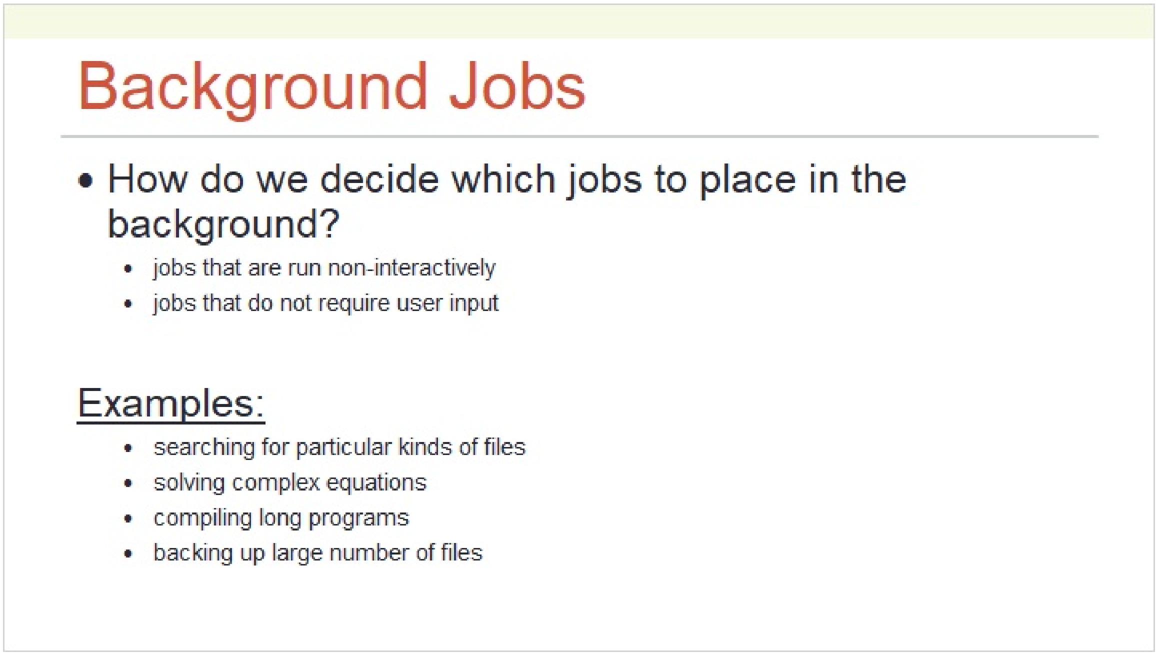
1.4 Job Control Terminology



Job Control Terminology

- Foreground job:
  - a job that has our immediate attention
  - user has to wait for job to complete
- Background job:
  - a job that the user does not wait for
  - it runs independently of user interaction
- Unix shells allow users to:
  - make jobs execute in the background,
  - move jobs from foreground to background,
  - determine their status, and terminate them

1.5 Background Jobs

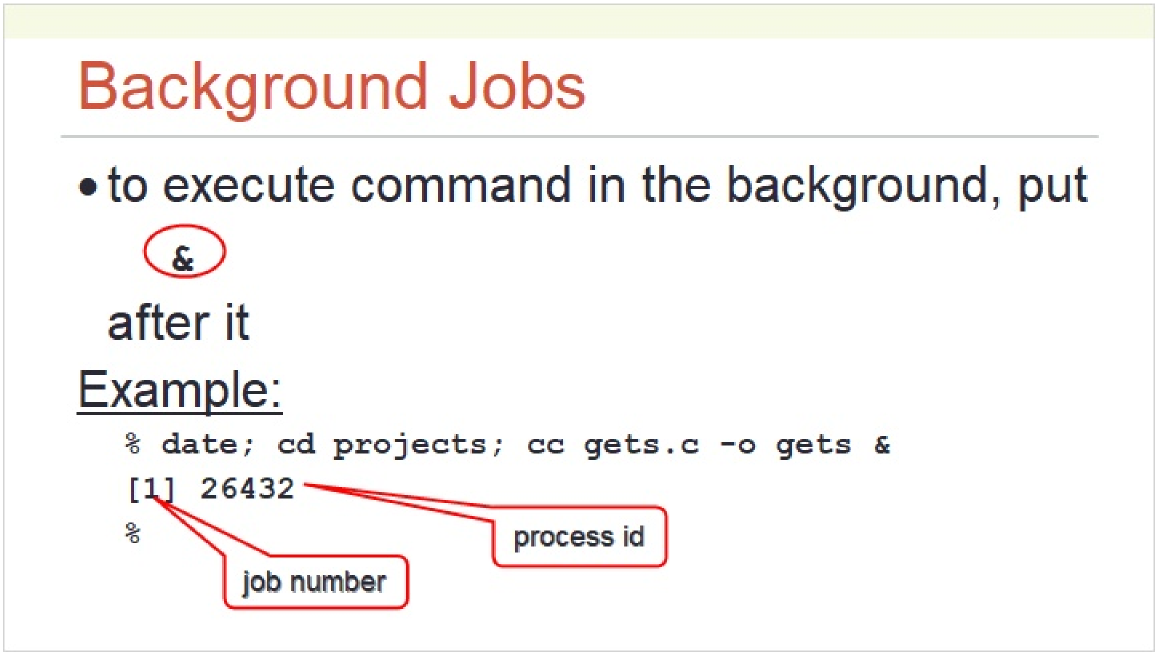


Background Jobs

- How do we decide which jobs to place in the background?
  - jobs that are run non-interactively
  - jobs that do not require user input

Examples:
- searching for particular kinds of files
- solving complex equations
- compiling long programs
- backing up large number of files

1.6 Background Jobs



Background Jobs

- to execute command in the background, put

  &

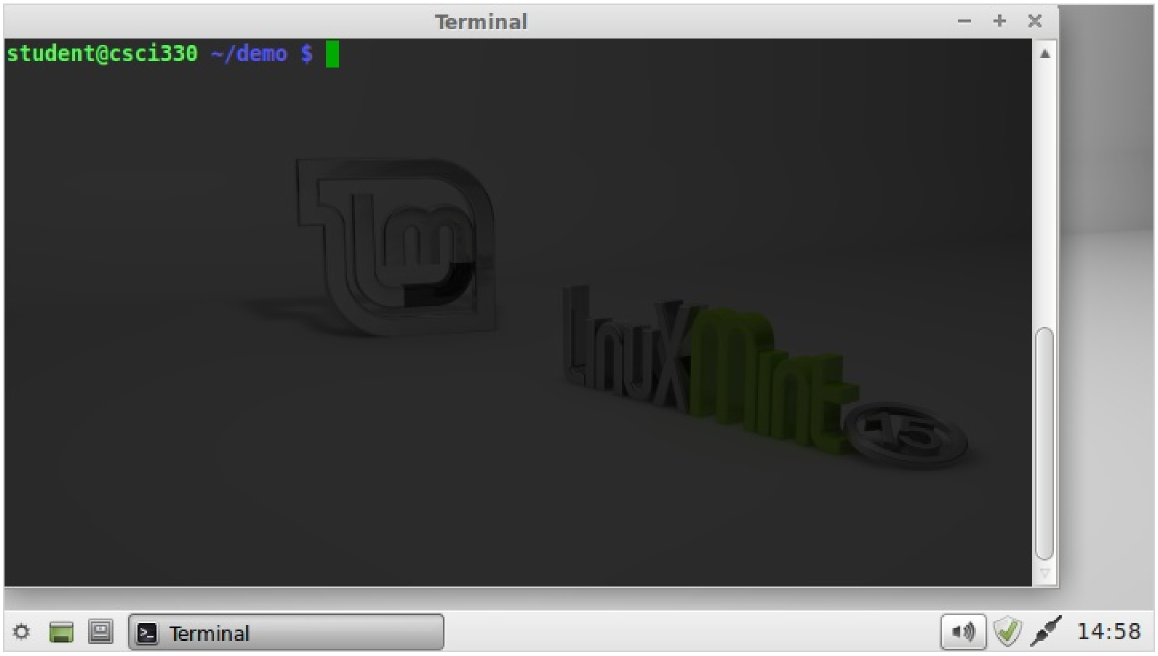
  after it

Example:

```
% date; cd projects; cc gets.c -o gets &
[1] 26432
%
```

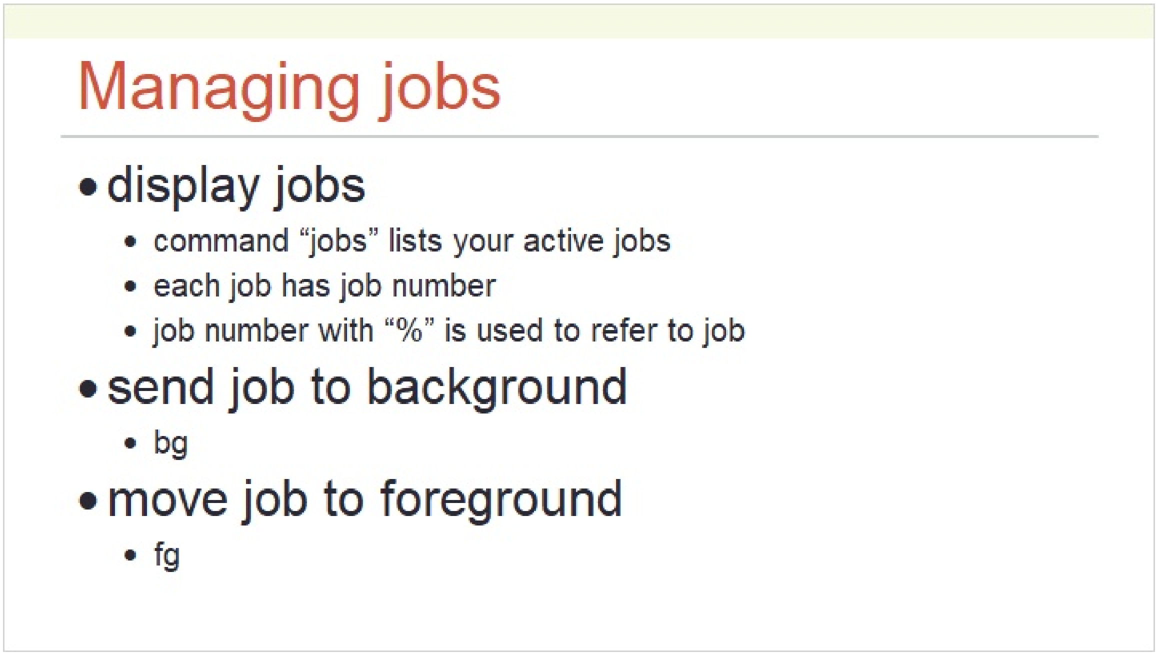
process id
job number

1.7 demonstration



```
student@csci330 ~/demo $ 
```

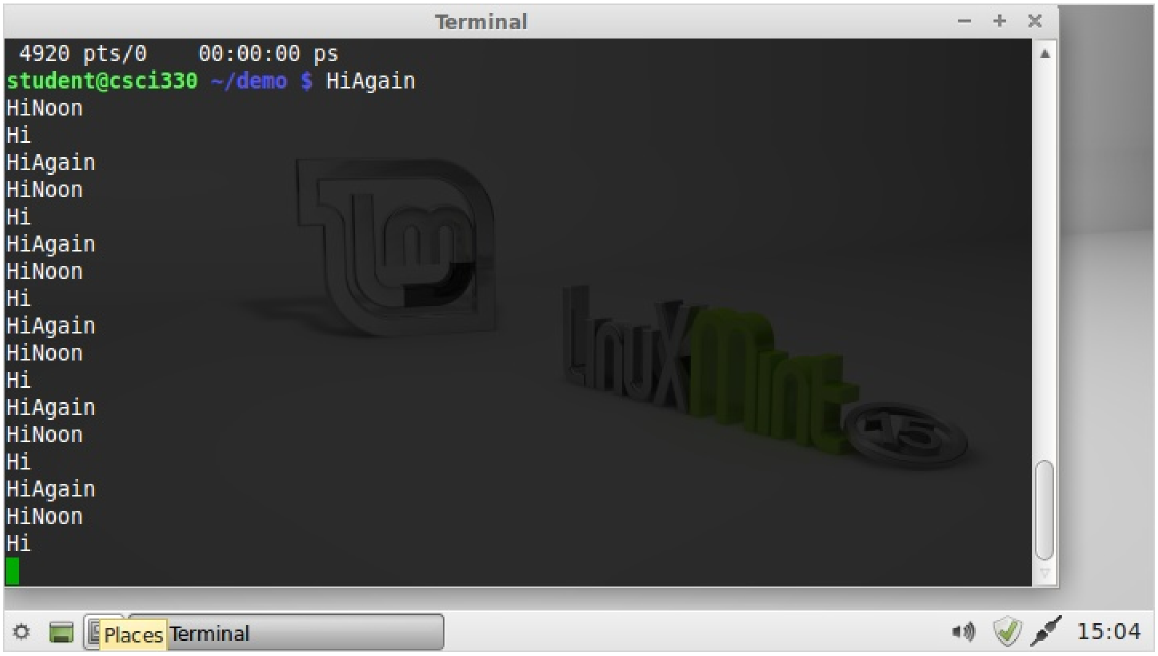
1.8 Managing jobs



Managing jobs

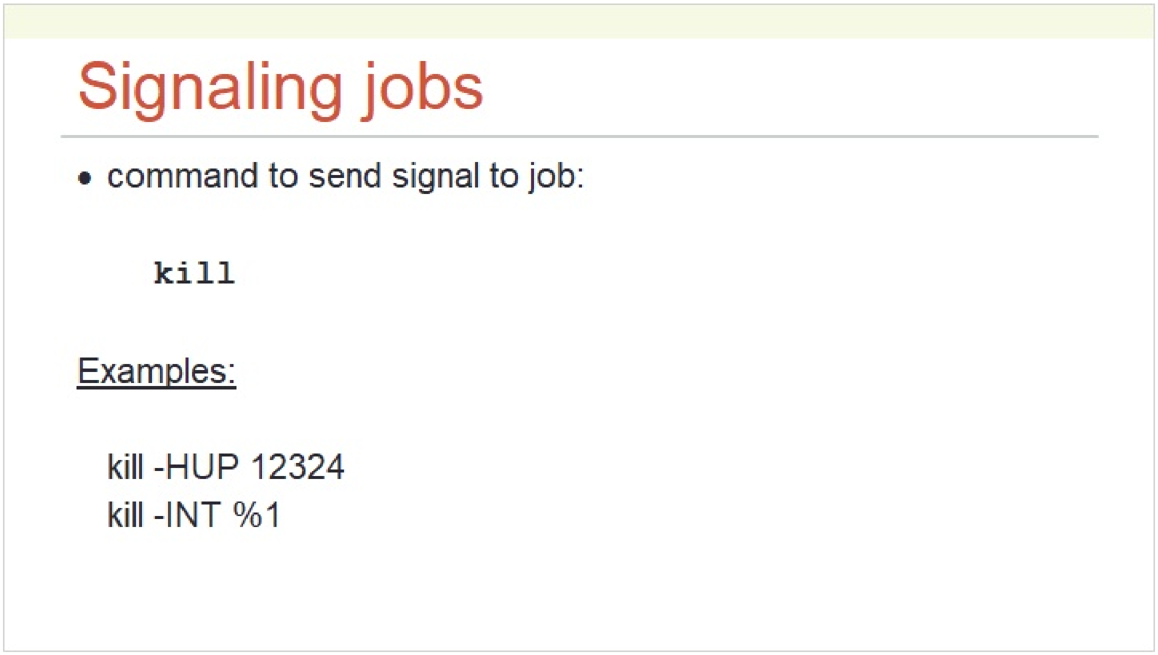
- display jobs
  - command “jobs” lists your active jobs
  - each job has job number
  - job number with “%” is used to refer to job
- send job to background
  - bg
- move job to foreground
  - fg

1.9 demonstration



```
 4920 pts/0    00:00:00 ps
student@csci330 ~/demo $ HiAgain
HiNoon
Hi
HiAgain
HiNoon
Hi
HiAgain
HiNoon
Hi
HiAgain
HiNoon
Hi
HiAgain
HiNoon
Hi
HiAgain
HiNoon
Hi
```

1.10 Signaling jobs



Signaling jobs

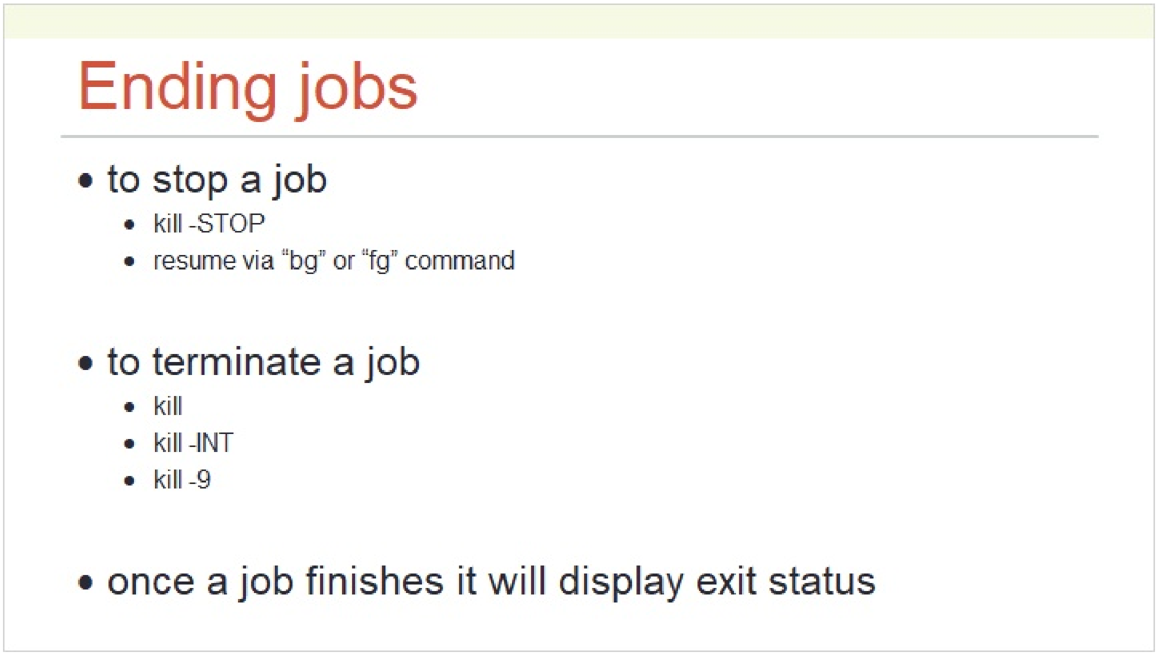
- command to send signal to job:

```
kill
```

Examples:

```
kill -HUP 12324
kill -INT %1
```

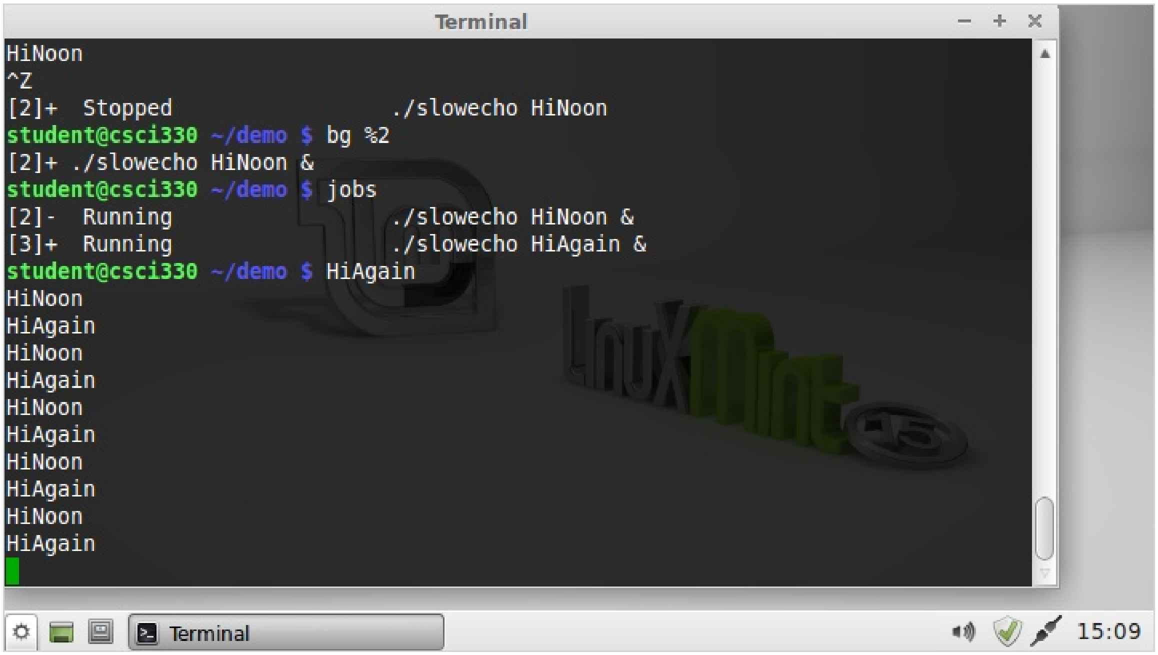
1.11 Ending jobs



Ending jobs

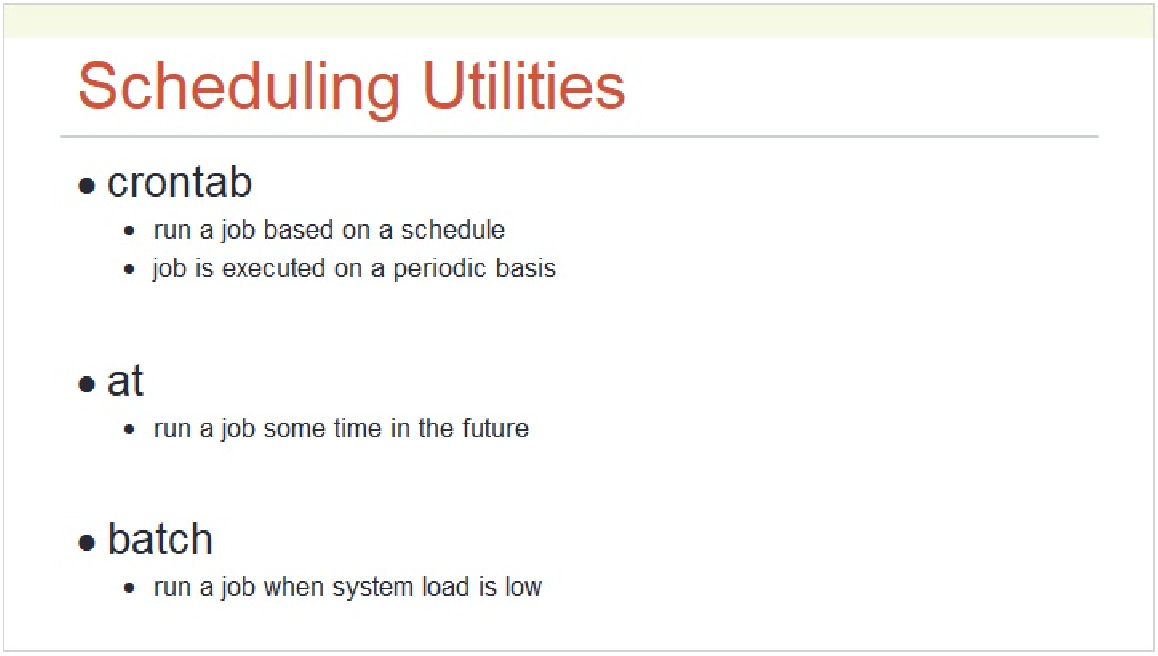
- to stop a job
  - kill -STOP
  - resume via “bg” or “fg” command
- to terminate a job
  - kill
  - kill -INT
  - kill -9
- once a job finishes it will display exit status

1.12 demonstration



```
HiNoon
^Z
[2]+  Stopped                 ./slowecho HiNoon
student@csci330 ~/demo $ bg %2
[2]+ ./slowecho HiNoon &
student@csci330 ~/demo $ jobs
[2]-  Running                 ./slowecho HiNoon &
[3]+  Running                 ./slowecho HiAgain &
student@csci330 ~/demo $ HiAgain
HiNoon
HiAgain
HiNoon
HiAgain
HiNoon
HiAgain
HiNoon
HiAgain
HiNoon
HiAgain
```

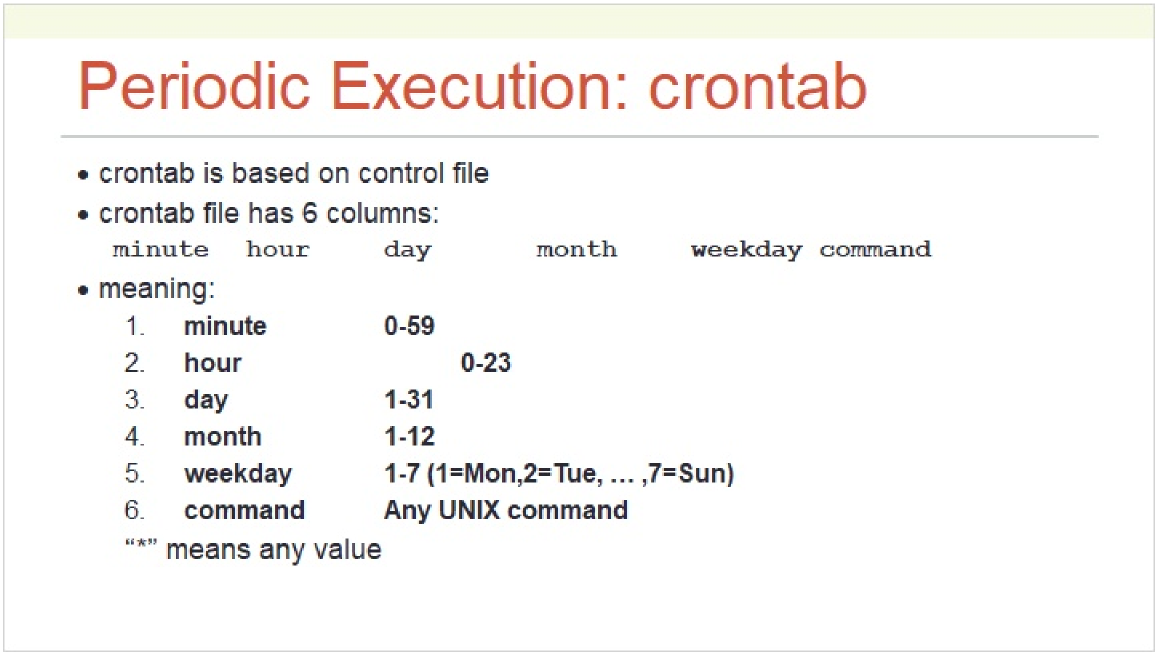
1.13 Scheduling Utilities



Scheduling Utilities

- crontab
  - run a job based on a schedule
  - job is executed on a periodic basis
- at
  - run a job some time in the future
- batch
  - run a job when system load is low

1.14 Periodic Execution: crontab



Periodic Execution: crontab

- crontab is based on control file
- crontab file has 6 columns:

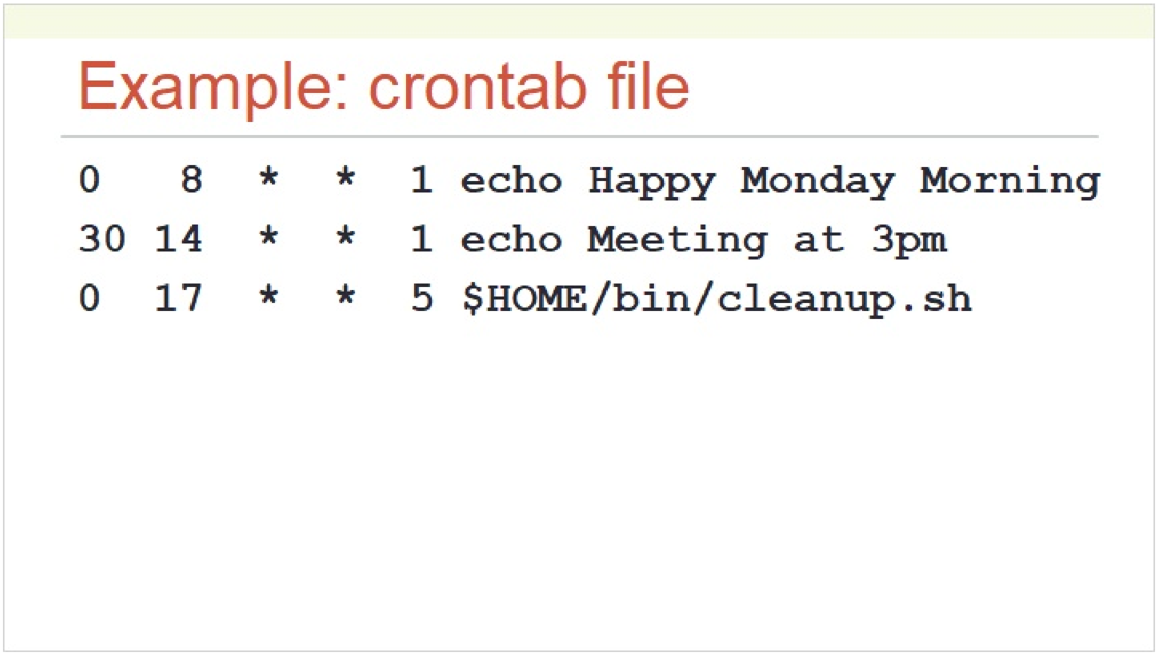
```
minute   hour     day        month     weekday  command
```

- meaning:

1.	**minute**	**0-59**
2.	**hour**	**0-23**
3.	**day**	**1-31**
4.	**month**	**1-12**
5.	**weekday**	**1-7 (1=Mon,2=Tue, ... ,7=Sun)**
6.	**command**	**Any UNIX command**

"*" means any value

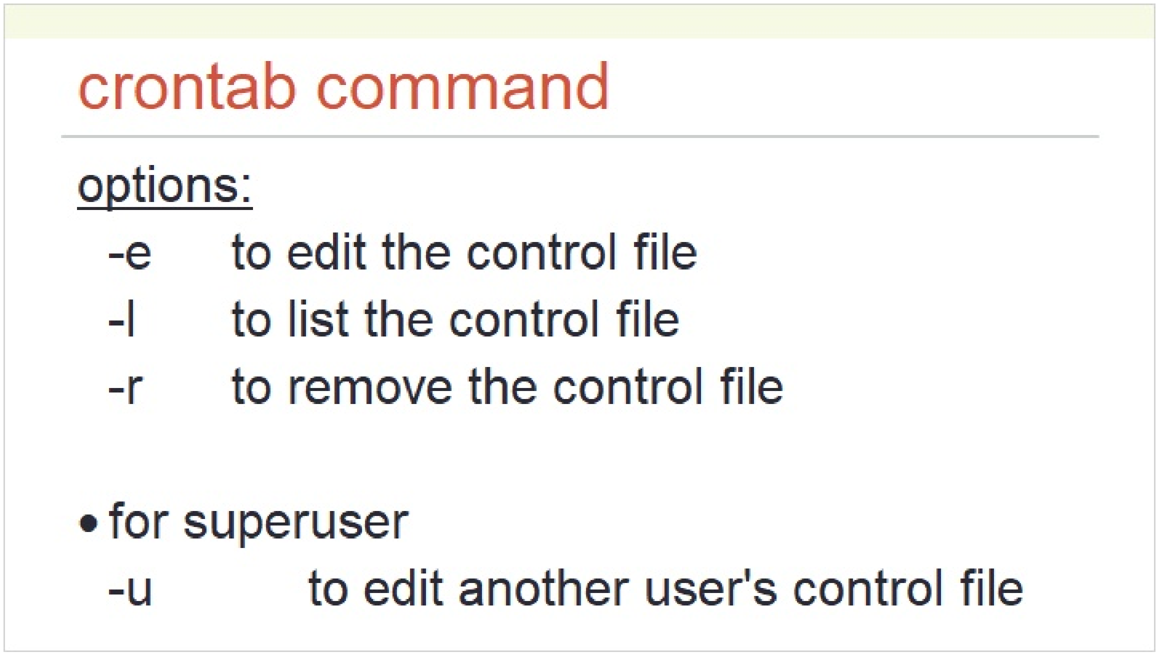
1.15 Example: crontab file



Example: crontab file

```
0    8  *   *   1 echo Happy Monday Morning
30  14  *   *   1 echo Meeting at 3pm
0   17  *   *   5 $HOME/bin/cleanup.sh
```

1.16 crontab command



crontab command

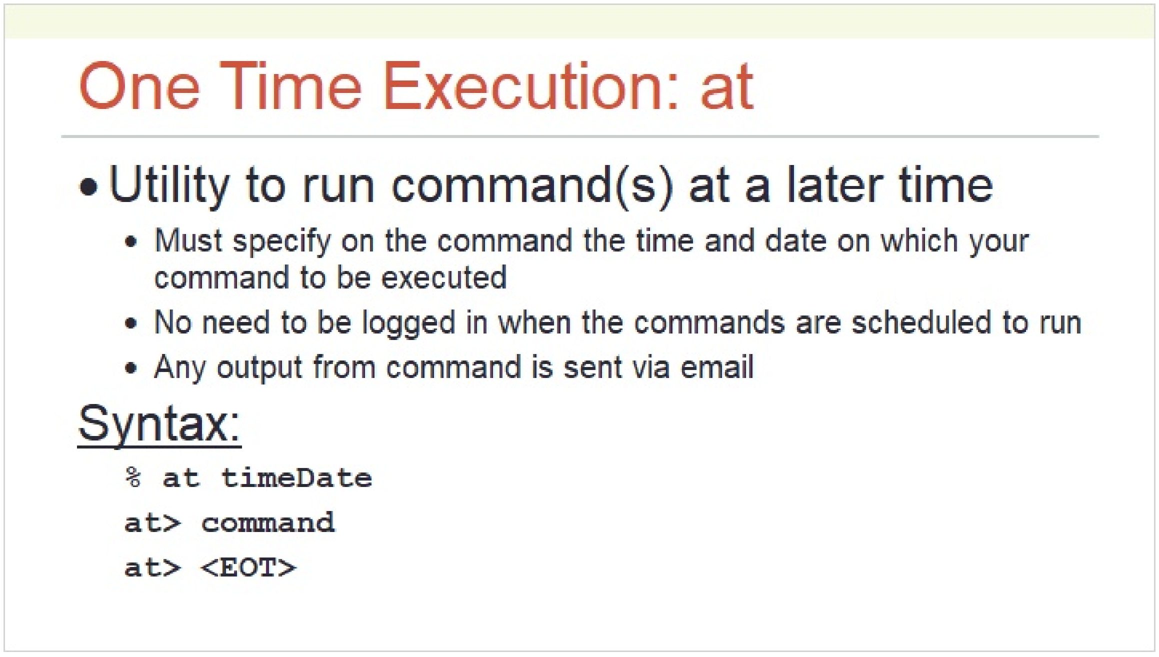
options:

-e	to edit the control file
-l	to list the control file
-r	to remove the control file

- for superuser

-u	to edit another user's control file

1.17 One Time Execution: at



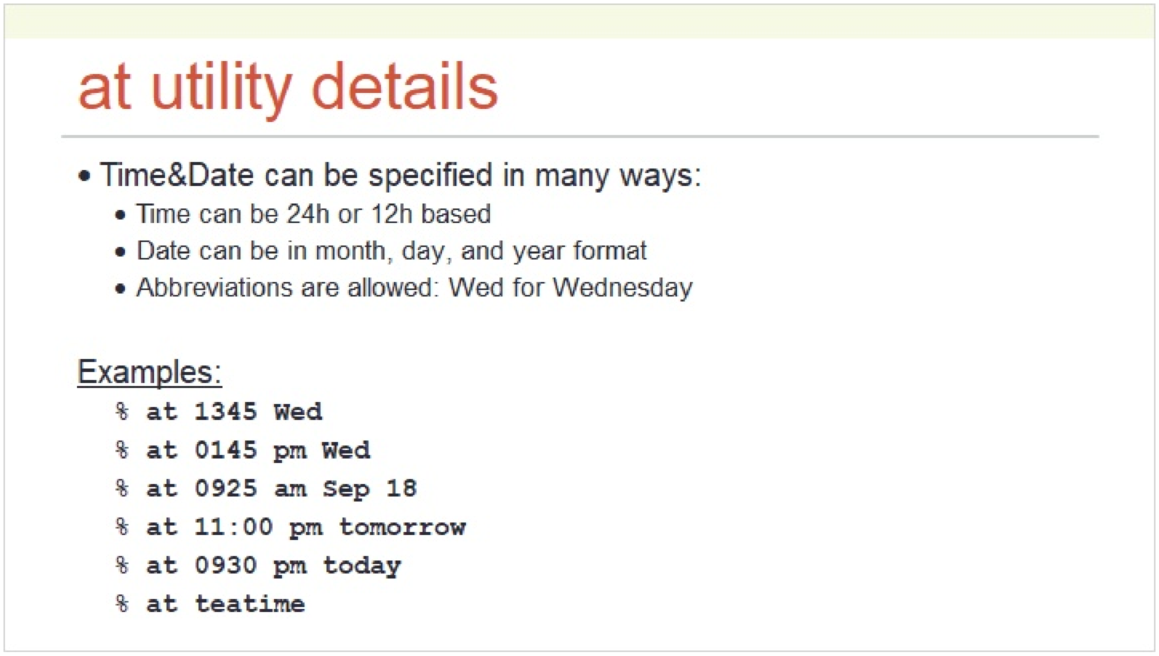
One Time Execution: at

- Utility to run command(s) at a later time
  - Must specify on the command the time and date on which your command to be executed
  - No need to be logged in when the commands are scheduled to run
  - Any output from command is sent via email

Syntax:

```
% at timeDate
at> command
at> <EOT>
```

1.18 at utility details



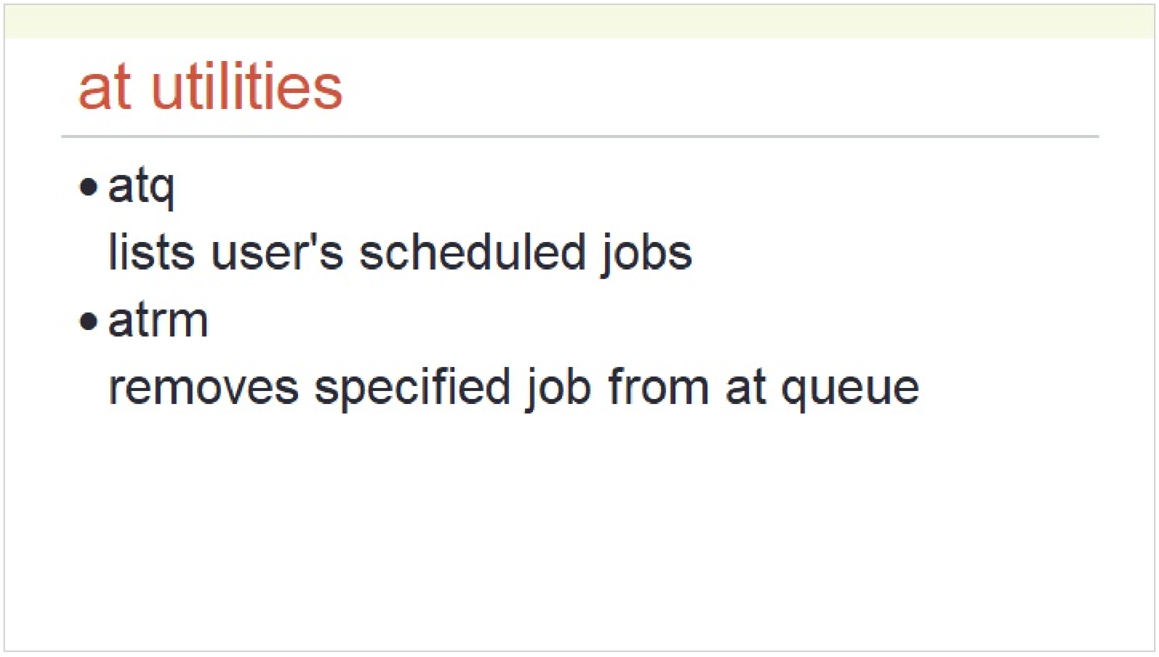
at utility details

- Time&Date can be specified in many ways:
  - Time can be 24h or 12h based
  - Date can be in month, day, and year format
  - Abbreviations are allowed: Wed for Wednesday

Examples:

```
% at 1345 Wed
% at 0145 pm Wed
% at 0925 am Sep 18
% at 11:00 pm tomorrow
% at 0930 pm today
% at teatime
```

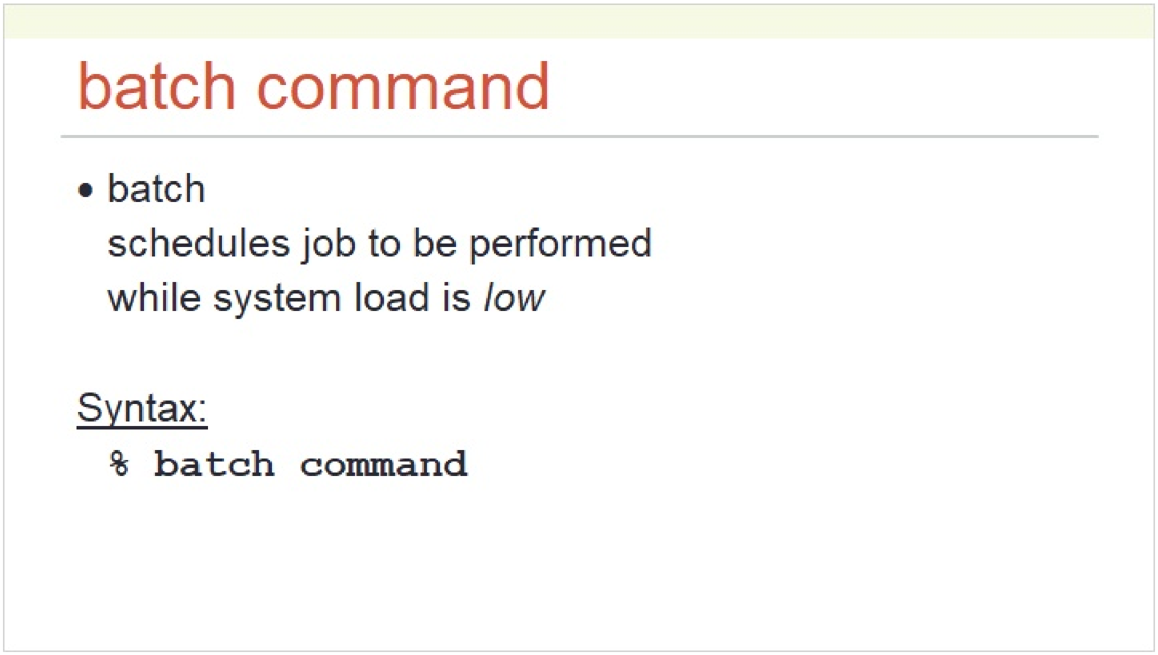
1.19 at utilities



at utilities

- atq
  lists user's scheduled jobs
- atrm
  removes specified job from at queue

1.20 batch command



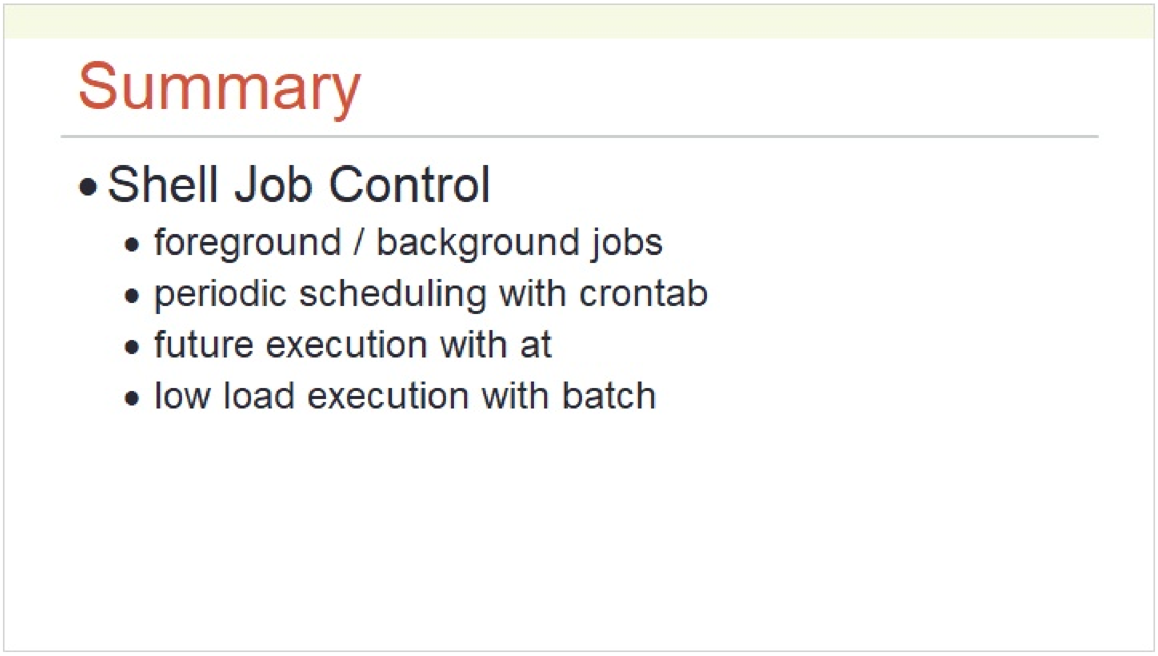
batch command

- batch
  schedules job to be performed
  while system load is *low*

Syntax:

```
% batch command
```

1.21 Summary



Summary

- Shell Job Control
  - foreground / background jobs
  - periodic scheduling with crontab
  - future execution with at
  - low load execution with batch