

# Some UNIX Commands

## at

Syntax:	at time [day] [file]
Description:	Provides ability to perform UNIX commands at some time in the future. At time <i>time</i> on day <i>day</i> , the commands in <i>filefile</i> will be executed.
Comment:	Often used to do time-consuming work during off-peak hours, or to remind yourself to do something during the day.  Other at-related commands:  <b>atq</b> - Dump the contents of the <i>at</i> event queue.  <b>atrm</b> - Remove <i>at</i> jobs from the queue.
Examples:	<pre># at 0300 calendar   mail john ^D #  # cat &gt; DOTHIS nroff -ms doc1 &gt;&gt; doc.fmt nroff -ms file2 &gt;&gt; doc.fmt spell doc.fmt &gt; SPELLerrs ^D # at 0000 DOTHIS</pre>

## cal

Syntax:	cal [month] year
Description:	Prints a calendar for the specified year or month. The <i>year</i> starts from year 0 [the calendar switched to <i>Julian</i> in 1752], so you typically must include the century
Comment:	Typical syntax:  cal 11 1997 cal 1998

## calendar

Syntax:	<code>calendar [-]</code>
Description:	You must set-up a file, typically in your home directory, called <i>calendar</i> . This is a database of events.
Comment:	Each event must have a date mentioned in it: Sept 10, 12/5, Aug 21 1998, ... Each event scheduled for the day or the next day will be listed on the screen.

## lpr

Syntax:	<code>lpr [OPTIONS] [FILE] [FILE] ...</code>
Description:	The files are queued to be output to the printer.
Comment:	On some UNIX systems, this command is called <i>lp</i> , with slightly different options. Both <i>lpr</i> and <i>lp</i> first make a copy of the file(s) to be printed in the spool directory. Then a <b>daemon</b> (background process) outputs the files to the printer device.

## pr

Syntax:	<code>pr [OPTIONS] [FILE] [FILE] ...</code>
Description:	The file is broken into pages, and each page is sent to <i>STDOUT</i> , headed with the date, filename, and page number.
Comment:	Unless specified otherwise, output goes to the screen. To send this output to a printer:  <code>pr THISFILE &gt; /dev/lp</code>

## cmp

Syntax:	<code>cmp [OPTIONS] FILE1 FILE2</code>
Description:	Two files are compared. If they are the same, no comment is generated. If they differ, the byte and line number of the first difference is displayed.
Comment:	Typical usage:  <code>\$ cmp edscript thisscript</code>

## split

Syntax:	<code>split [-n] [FILE [NAME]]</code>
Description:	<i>Split</i> reads the <i>FILE</i> and writes it in <i>n</i> -line pieces (by default, <i>n</i> is 1000), as many as necessary onto a set of output files. The first output file is the <i>NAME</i> with <i>aa</i> appended, or <i>x</i> with <i>aa</i> appended if no <i>NAME</i> is specified.
Example:	<pre>\$ ls THISFILE \$ split -4 THISFILE \$ ls THISFILE xaa xab xac \$</pre>

## tail

Syntax:	<code>tail ±NUMBER [OPTIONS] [FILE]</code>
Description:	Prints the file on the screen + the number of lines from the beginning of the file, or - the number of lines from the end.
Example:	<pre>\$ cat &gt; test THIS IS A TEST ^D \$ tail -2 test IS A TEST \$</pre>

## file

Syntax:	<code>file FILENAME</code>
Description:	Returns the type of information contained in the file <i>FILENAME</i> . File tries to guess the content based on the first few characters and a database of known file types. <i>It is not always accurate!</i>
Sample Responses:	<pre>a.out : Executable not stripped prog.c : C program text adir : Directory edscript : English text test : ASCII text</pre>

## echo

Syntax:	echo [-n] [arg] [arg] ...
Description:	<i>echo</i> writes its arguments, separated by blanks and terminated by a newline (unless the <i>-n</i> option is used), to the standard output.
Example:	\$ echo THIS IS A BUNCH OF ARGS THIS IS A BUNCH OF ARGS \$
Comment:	<i>echo</i> is heavily used in shell scripts to tell the human what's going on or to prompt for an input.

## sleep

Syntax:	sleep time
Description:	<i>Sleep</i> suspends execution for <i>time</i> seconds.
Comment:	<i>Sleep</i> is typically combined with other commands to delay their execution.  \$ (sleep 105; command)

## ps

Syntax:	ps [OPTIONS]
Description:	<i>ps</i> shows that processes that you invoked from your terminal.
Example:	Let's look at a process in the background:  \$ ps PID TTY TIME CMD 19 co 0:04 -sh \$ (sleep 105; ls) & \$ ps PID TTY TIME CMD 19 co 0:04 -sh 478 co 0:00 -sh 479 co 0:00 sleep 105 \$

## Background

□ You can send any UNIX command into the background to execute.

□ Your prompt comes right back, allowing you to do other work in the foreground.

□ Example:

```
$ (sleep 105; ls) &
373
$
```

There is now a process in the background with process ID = 373, which will wait for 105 seconds, then print a directory listing to the screen.

## kill

Syntax:	kill [OPTIONS] PROCESSID
Description:	Kills a process in the background. You may only kill your own processes. The superuser can kill any user's process. Can be used to send other <i>signals</i> to processes.
Example:	<pre>\$ ps PID TTY TIME CMD 19 co 0:04 -sh \$ (sleep 300; ls)&amp; 433 \$ kill -9 433 433 killed \$ ps PID TTY TIME CMD 19 co 0:05 -sh 434 co 0:00 sleep 300</pre>
Comment:	<i>Note: kill -9 is an unconditional kill ("kill with extreme prejudice").</i>