


- Linux History
- Distributions Debian / Ubuntu
- Installation (Single/Dual mode – Virtual Box – Portable)
- Command line (Terminal)



Open-Source Operating Systems



- Operating systems made available in source-code format rather than just binary **closed-source**.
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has “copy left” **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD** (*Berkeley Software Distribution*) **UNIX** (including core of **Mac OS X**), a  any more
- Can use VMM (Virtual Machine Management) like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - <http://www.virtualbox.com>)
Use to run guest operating systems for exploration

is programming that's written to the read-only memory (ROM) of a computing device, which is added at the time of manufacturing, is used to run user programs on the device. (IBM prefers the term microcode)



Traditional UNIX Systems

- Were developed at Bell Labs and became operational on a PDP-7 (Microcomputer 1965) in 1970. (Written by Assembly Language – Ken Thompson)
- Incorporated many ideas from Multics (Multiplexed Information and Computing Service) is a timesharing operating system begun in 1965 and used until 2000.
- PDP-11(1970 -1990) was a milestone because it first showed that UNIX would be an OS for all computers.



- Next milestone was rewriting UNIX in the programming language C
 - demonstrated the advantages of using a high-level language for system code (ken Thompson & Dennis Ritchie C Author) 1973
- Was described in a technical journal for the first time in 1974
- First widely available version outside Bell Labs was Version 6 in 1976
- Version 7, released in 1978 is the ancestor of most modern UNIX systems
- Most important of the non-AT&T systems was UNIX BSD (Berkeley Software Distribution)



UNIX = UNICS

Uniplexed Information and Computing System



Unix Structure

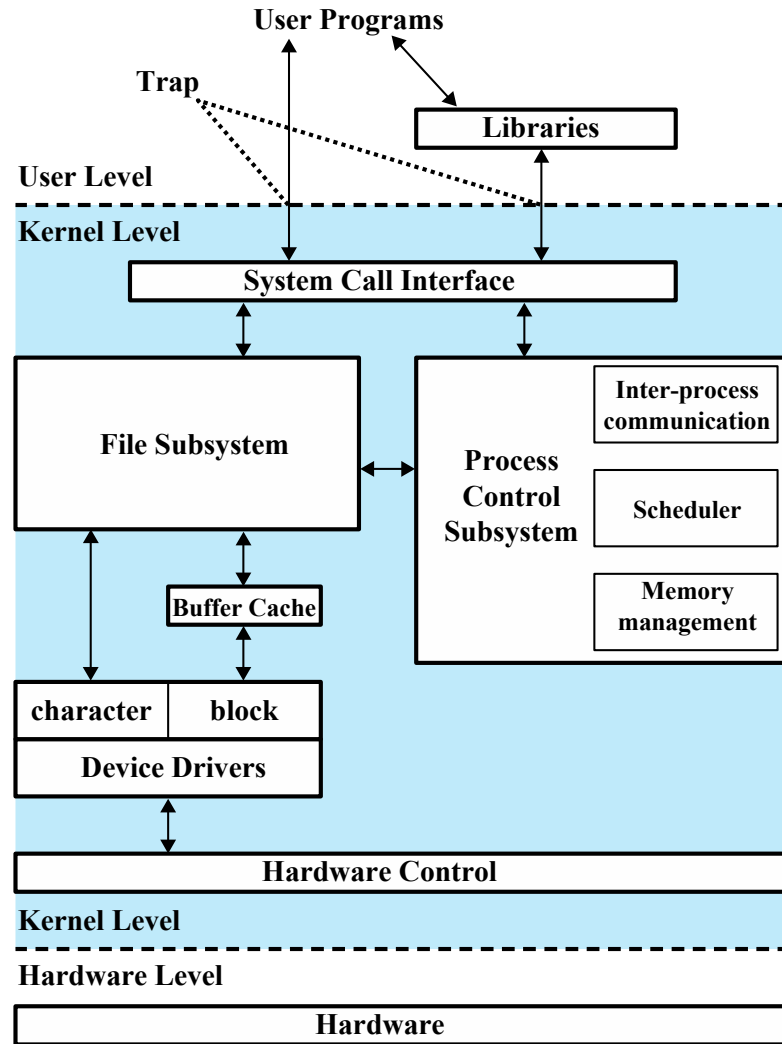


Figure 2.16 Traditional UNIX Kernel

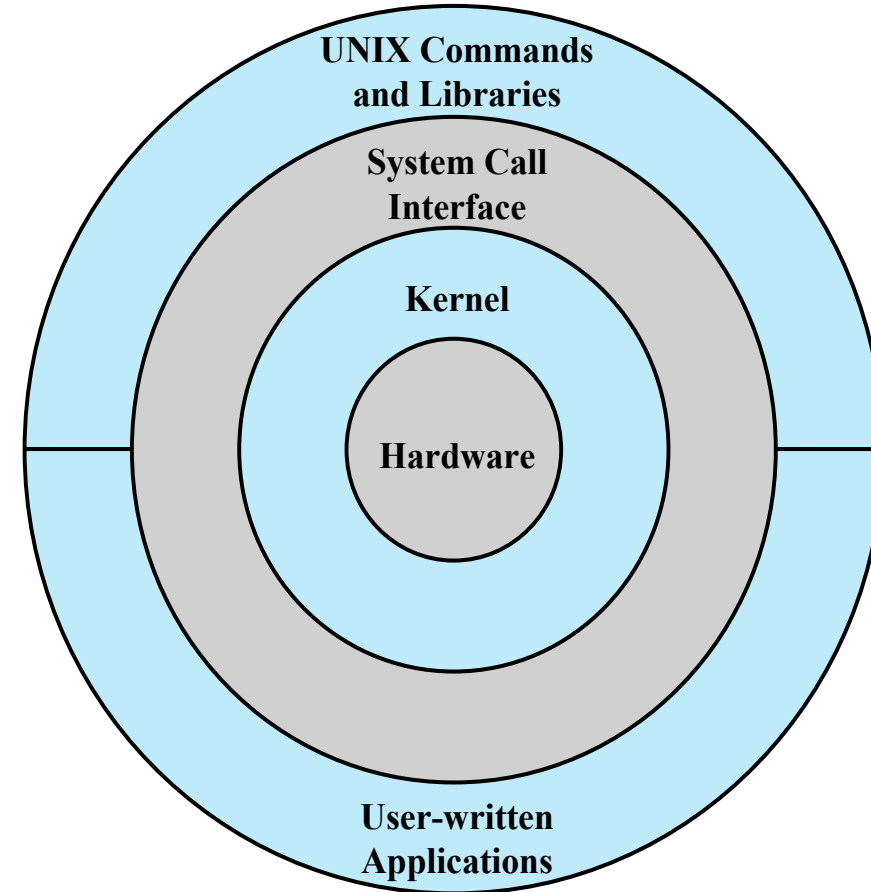


Figure 2.15 General UNIX Architecture

- Started out as a UNIX variant for the IBM PC
- Linus Torvalds, a Finnish student of computer science, wrote the initial version
- Linux was first posted on the Internet in 1991
- Today it is a full-featured UNIX system that runs on several platforms & Distributions of linux (Red Hat, Solaris, Debian, Ubuntu, Fedora, etc.)
- Is free and the source code is available
- Key to success has been the availability of free software packages
- Highly modular and easily configured



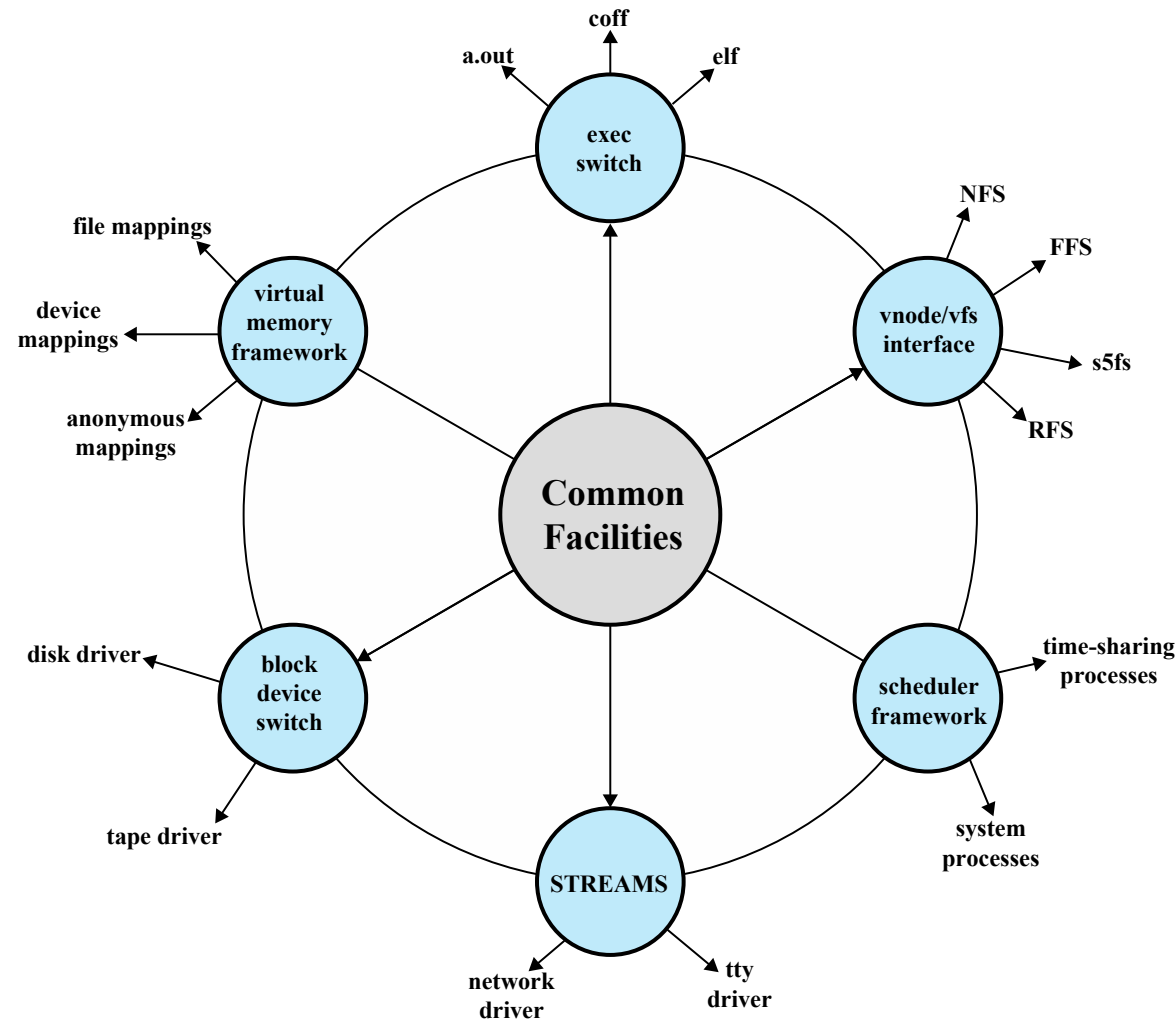


Figure 2.17 Modern UNIX Kernel [VAHA96]

User Mode

- user program executes in user mode
- certain areas of memory are protected from user access
- certain instructions may not be executed

Kernel Mode

- monitor executes in kernel mode
- privileged (higher priority) instructions may be executed
- protected areas of memory may be accessed



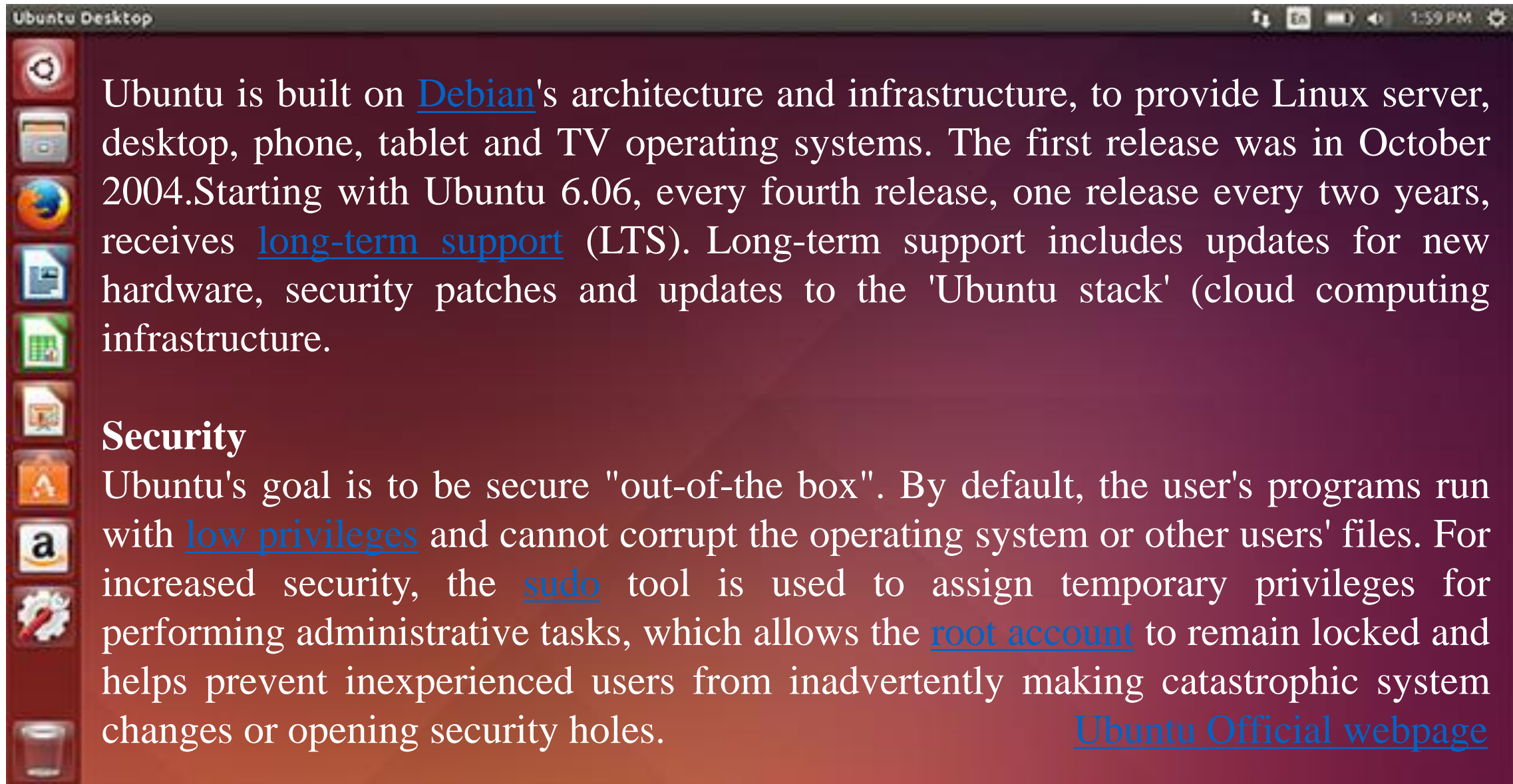


The Debian Project was first announced in 1993 by **Ian Murdock**, Debian 0.01 was released on September 15, 1993, and the first stable release was made in 1996.

The Debian stable [release branch](#) is one of the most popular for [personal computers](#) and [network servers](#), and has been used as a base for several other distributions.

[Debian Official Webpage](#)



A screenshot of an Ubuntu Desktop window. The title bar says "Ubuntu Desktop". The top right corner shows system icons for network, sound, and power, along with the time "1:59 PM". The left sidebar contains several application icons: Dash, Home Folder, Firefox, LibreOffice Writer, LibreOffice Calc, LibreOffice Impress, LibreOffice Draw, and a terminal. The main content area has a dark purple background with white text.

Ubuntu is built on [Debian](#)'s architecture and infrastructure, to provide Linux server, desktop, phone, tablet and TV operating systems. The first release was in October 2004. Starting with Ubuntu 6.06, every fourth release, one release every two years, receives [long-term support](#) (LTS). Long-term support includes updates for new hardware, security patches and updates to the 'Ubuntu stack' (cloud computing infrastructure).

Security

Ubuntu's goal is to be secure "out-of-the box". By default, the user's programs run with [low privileges](#) and cannot corrupt the operating system or other users' files. For increased security, the [sudo](#) tool is used to assign temporary privileges for performing administrative tasks, which allows the [root account](#) to remain locked and helps prevent inexperienced users from inadvertently making catastrophic system changes or opening security holes.

[Ubuntu Official webpage](#)



Downloading & Preparing ISO Memory Stick/ DVD

- [Debian](#)
- [Ubuntu](#) LTS
- [AOMEI](#) Partition Assistant
- Bootable Memory Stick ([Rufus](#))
- Bootable DVD ([IsoCreator](#), [Free ISO Burner](#))
- [Virtual Box](#)



1- Single Mode (only one OS) easy straight forward

2- Parallel Dual Mode (Win / Linux) depending on system hardware VGA, RAM, CPU core, Boot BIOS (basic input/output system) / UEFI (Unified Extensible Firmware Interface) (precisely updated UEFI)

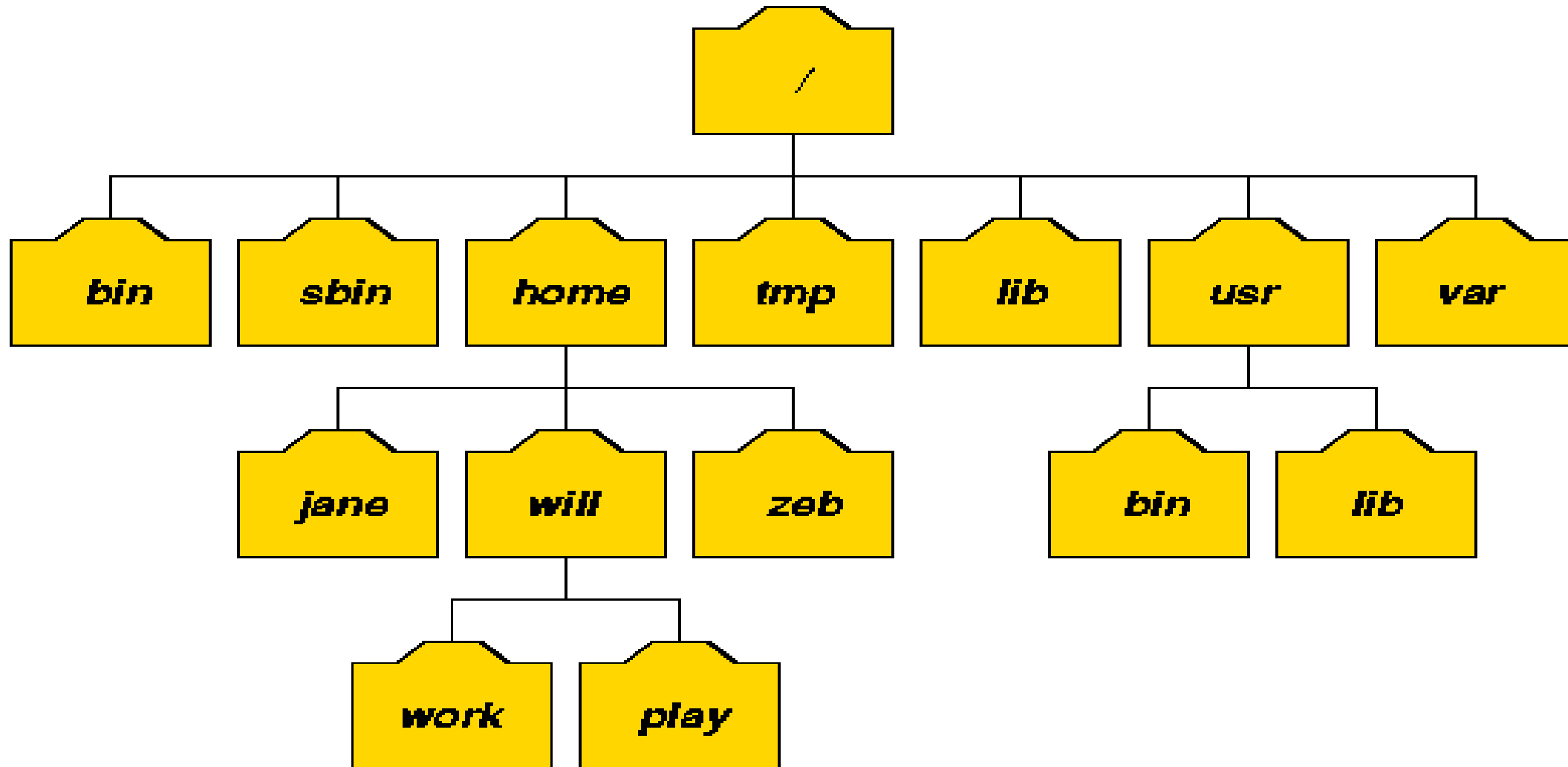
Installation Error Requires Community following for solutions.

3- Virtual Box Dual Mode window main platform

4- Portable Mode Memory Stick



Typical UNIX Directory Structure



Typical UNIX Directory Structure

<u>Directory</u>	<u>Typical Contents</u>
/	The "root" directory
/bin	Essential low-level system utilities
/usr/bin	Higher-level system utilities and application programs
/sbin	Superuser system utilities (for performing system administration tasks)
/lib	Program libraries (collections of system calls that can be included in programs by a compiler) for low-level system utilities
/usr/lib	Program libraries for higher-level user programs
/tmp	Temporary file storage space (can be used by any user)
/home or /homes	User home directories containing personal file space for each user. Each directory is named after the login of the user.
/etc	UNIX system configuration and information files
/dev	Hardware devices
/proc	A pseudo-filesystem which is used as an interface to the kernel. Includes a sub-directory for each active program (or process).



Linux Filesystem Categories

Every item stored in a UNIX filesystem belongs to one of four types:

1 - Ordinary files can contain text, data, or program information.

2 - Directories are containers or folders that hold files, and other directories.

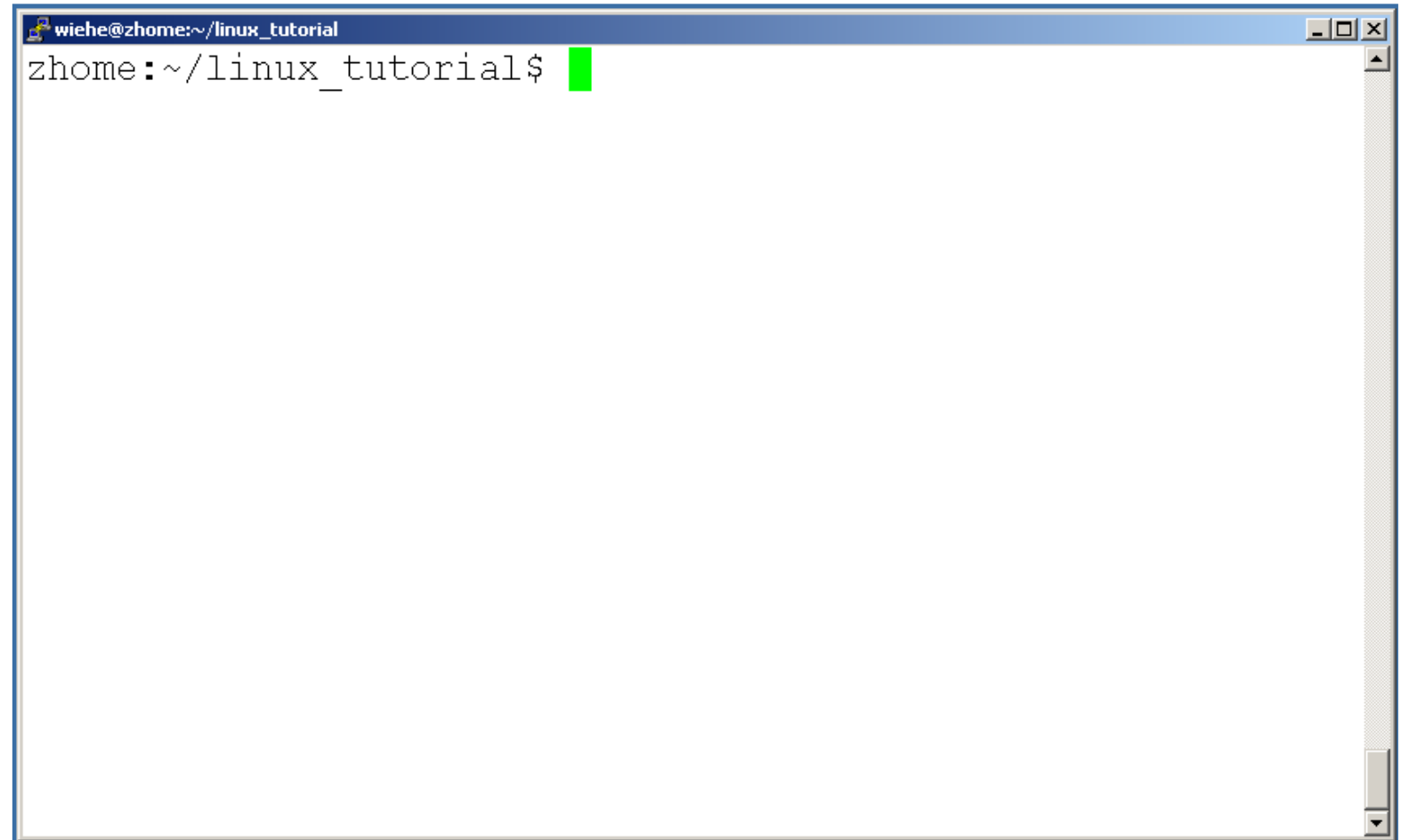
3 - Devices To provide applications with easy access to hardware devices, UNIX allows them to be used in much the same way as ordinary files. There are two types of devices in UNIX - **block-oriented** devices which transfer data in blocks (e.g. hard disks) and **character-oriented** devices that transfer data on a byte-by-byte basis (e.g. modems and dumb terminals).

4 - Links A link is a pointer to another file. There are two types of links - a **hard link** to a file is indistinguishable from the file itself. A **soft link** (or symbolic link) provides an indirect pointer or shortcut to a file.



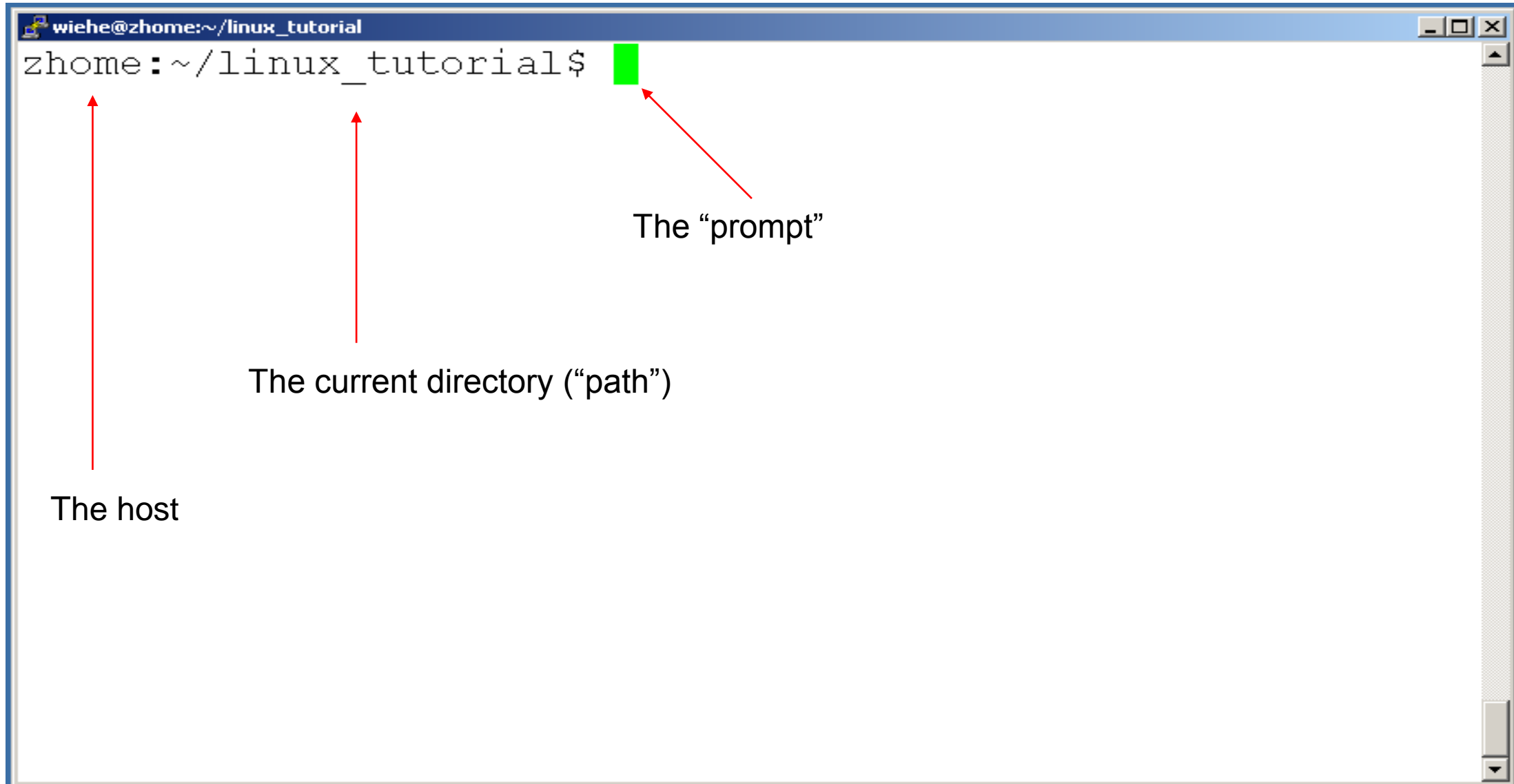
Open up a terminal:

- Ctrl + Alt + T
- Double click selection



Linux Terminal Info

19



A screenshot of a Linux terminal window. The title bar at the top reads "wiehe@zhome:~/linux_tutorial". The terminal content shows the prompt "zhome:~/linux_tutorial\$". A green cursor is positioned at the end of the prompt. Three red arrows point from text labels to parts of the prompt: one from "The host" to "wiehe", one from "The current directory ('path')" to "~/linux_tutorial", and one from "The 'prompt'" to the green cursor.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$
```

The host

The current directory ("path")

The "prompt"

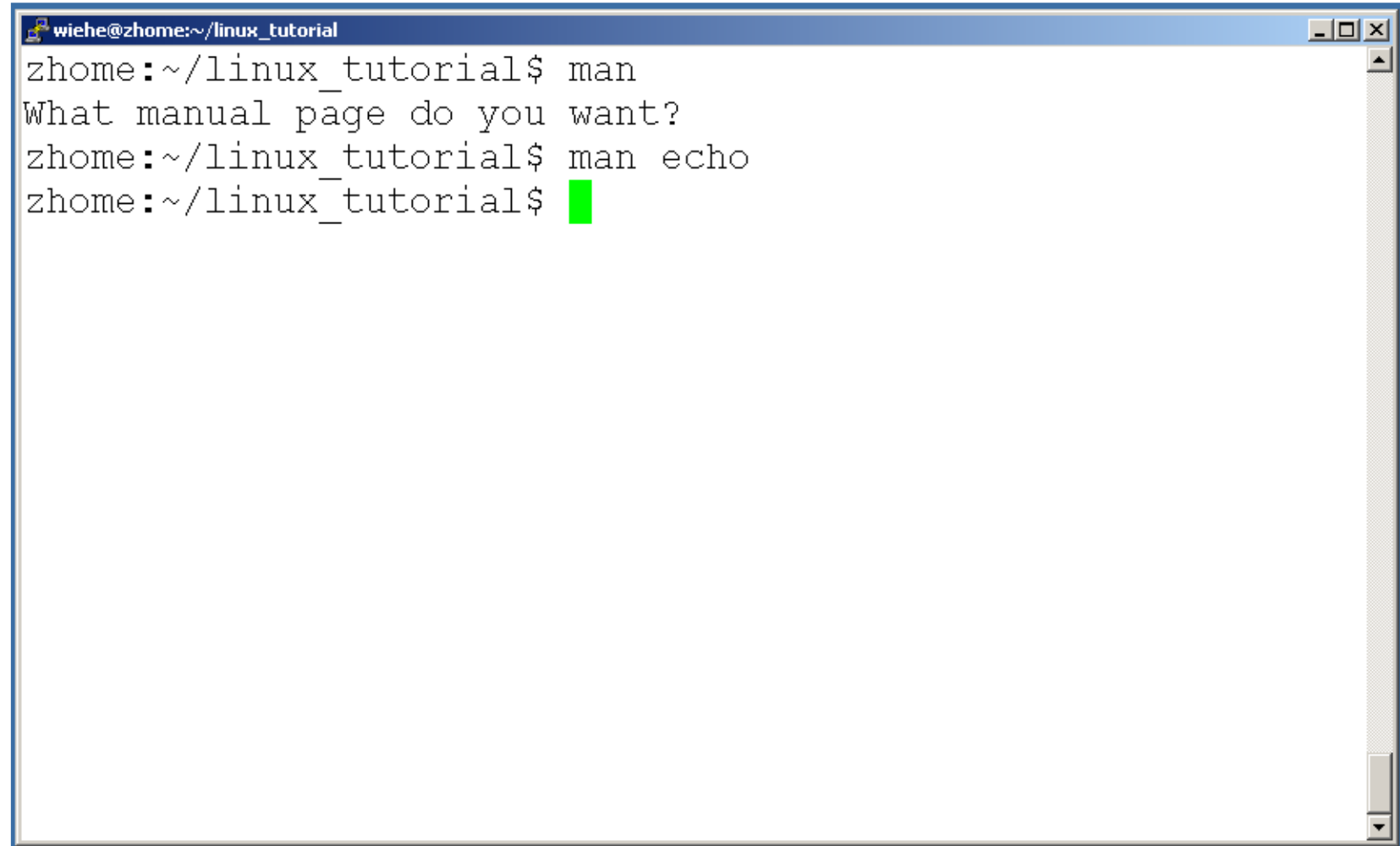


- After logging in, Linux/Unix starts another program called the **shell**
- The shell interprets commands the user types and manages their execution
 - The shell communicates with the internal part of the operating system called the **kernel**
 - The most popular shells are: tcsh, csh, korn, and bash
 - The differences are most times subtle
 - For this tutorial, we are using bash
- Shell commands are **CASE SENSITIVE!**



Linux man

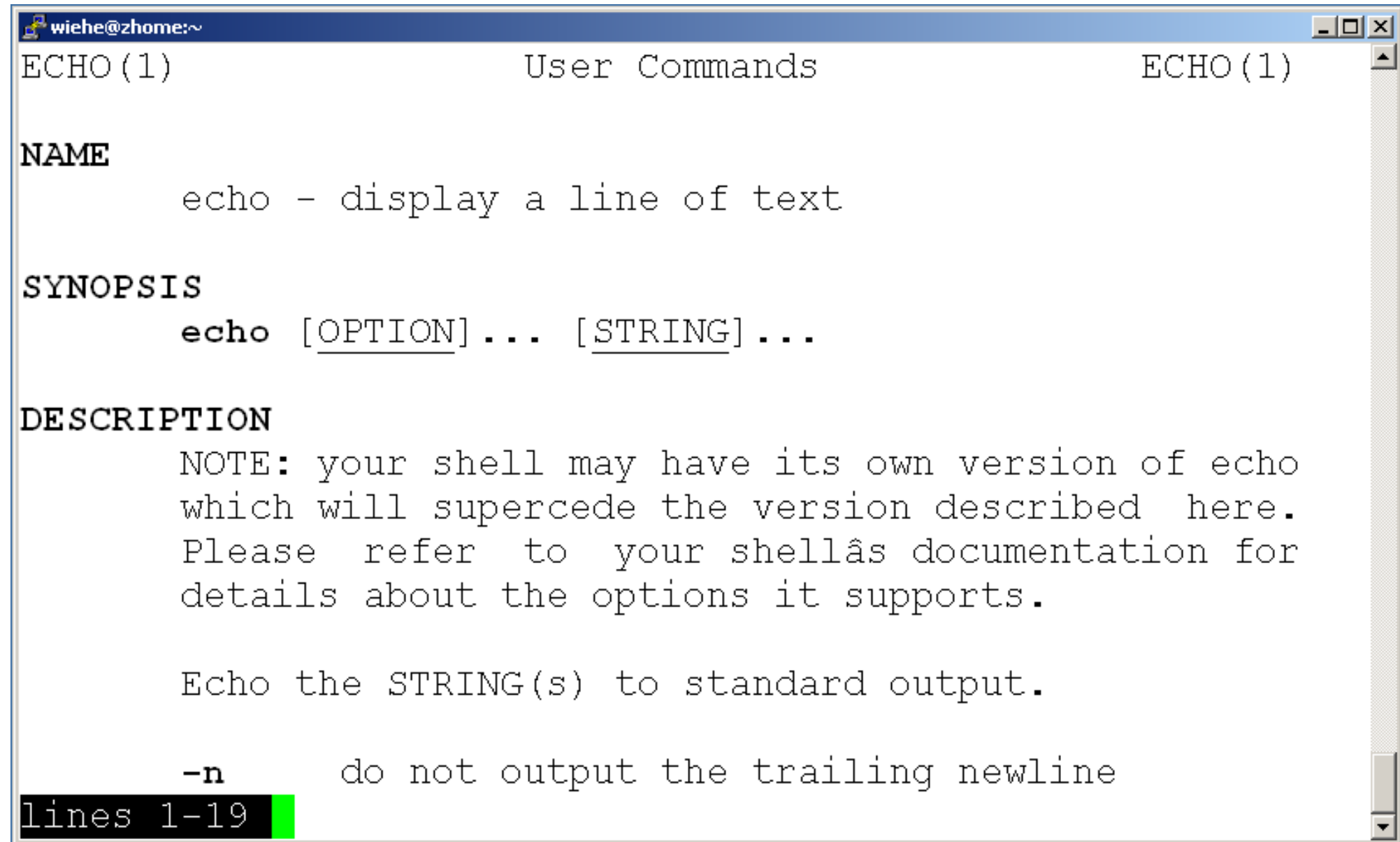
Whenever you need help with a command type “**man**” and the command name



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ man
What manual page do you want?
zhome:~/linux_tutorial$ man echo
zhome:~/linux_tutorial$
```



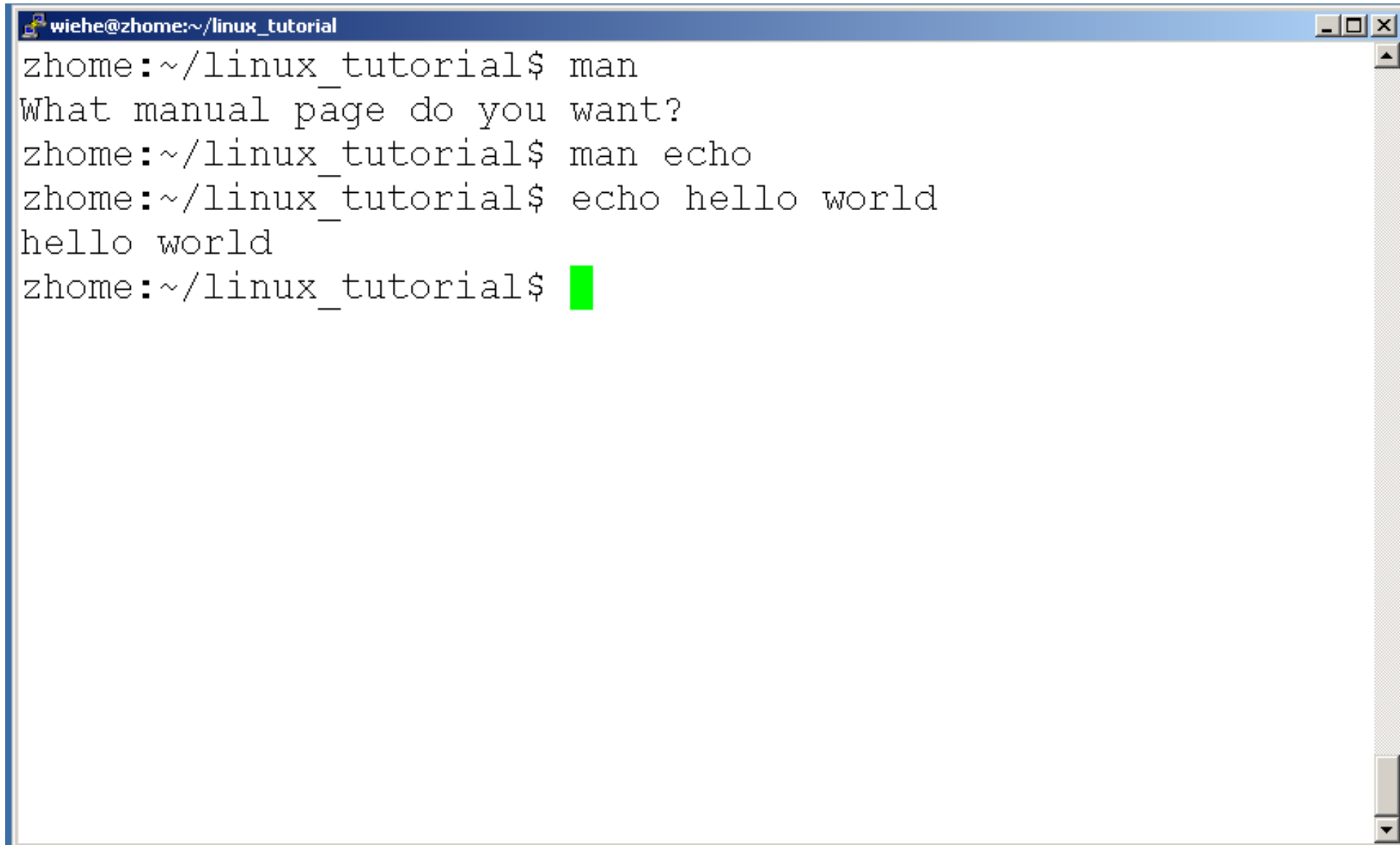
Man example



```
wiehe@zhome:~  
ECHO (1)                                User Commands                                ECHO (1)  
  
NAME  
    echo - display a line of text  
  
SYNOPSIS  
    echo [OPTION]... [STRING]...  
  
DESCRIPTION  
    NOTE: your shell may have its own version of echo  
    which will supercede the version described here.  
    Please refer to your shell's documentation for  
    details about the options it supports.  
  
    Echo the STRING(s) to standard output.  
  
    -n      do not output the trailing newline  
lines 1-19
```



echo example

A terminal window titled 'wiehe@zhome:~/linux_tutorial' with standard window controls. The terminal shows a sequence of commands and their outputs: 'man' is entered, followed by the prompt 'What manual page do you want?'; 'man echo' is entered; 'echo hello world' is entered, resulting in the output 'hello world'; and finally, the prompt 'zhome:~/linux_tutorial\$' is shown with a green cursor.

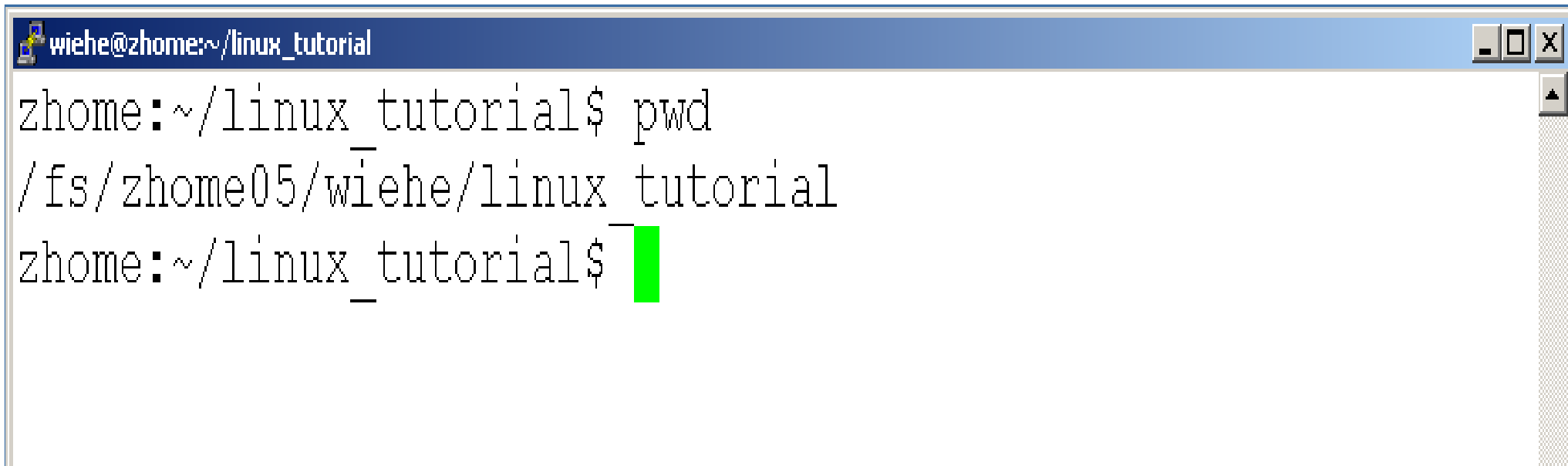
```
wiehe@zhome:~/linux_tutorial$ man
What manual page do you want?
zhome:~/linux_tutorial$ man echo
zhome:~/linux_tutorial$ echo hello world
hello world
zhome:~/linux_tutorial$
```



- **pwd (print [current] working directory)**

To find your current path use “pwd”

\$ pwd (Enter) output /usr/bin



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux tutorial$ pwd
/fs/zhome05/wiehe/linux tutorial
zhome:~/linux_tutorial$
```

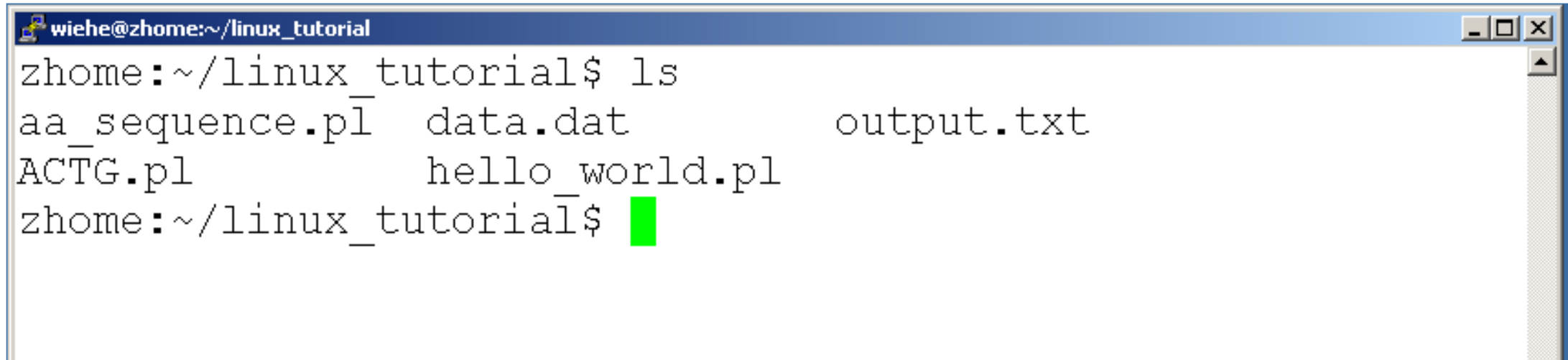


- **ls (list directory)**

To list the files in the current directory use “ls”

\$ ls (Enter) out put

bin dev home mnt share usr var boot etc lib proc sbin tmp vol



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat          output.txt
ACTG.pl         hello_world.pl
zhome:~/linux_tutorial$
```



ls has many options

- -a (all)
- -l long list (displays lots of info)
- -t sort by modification time
- -S sort by size
- -h list file sizes in human readable format
- -r reverse the order

“man ls” for more options

Options can be combined: “ls -ltr” , “ls -al”



Linux Terminal Commands

27

Filetype and permissions	Number of enclosed files	Owner	Group	Size (K)	Modification date and time	File name
drwxr-xr-x	31	andrea	andrea	4096	2013-01-18 08:38	.
drwxr-xr-x	3	root	root	4096	2010-10-31 20:06	..
-rw-----	1	andrea	andrea	3695	2013-01-19 13:01	.bash_history
-rw-r--r--	1	andrea	andrea	220	2010-10-31 20:06	.bash_logout
-rw-r--r--	1	andrea	andrea	3103	2010-10-31 20:06	.bashrc
drwx-----	5	andrea	andrea	4096	2013-01-18 08:38	.cache
drwxr-xr-x	12	andrea	andrea	4096	2011-05-03 17:01	.config
drwx-----	3	andrea	andrea	4096	2010-10-31 20:10	.dbus
drwxr-xr-x	6	andrea	andrea	4096	2011-03-23 06:23	Desktop

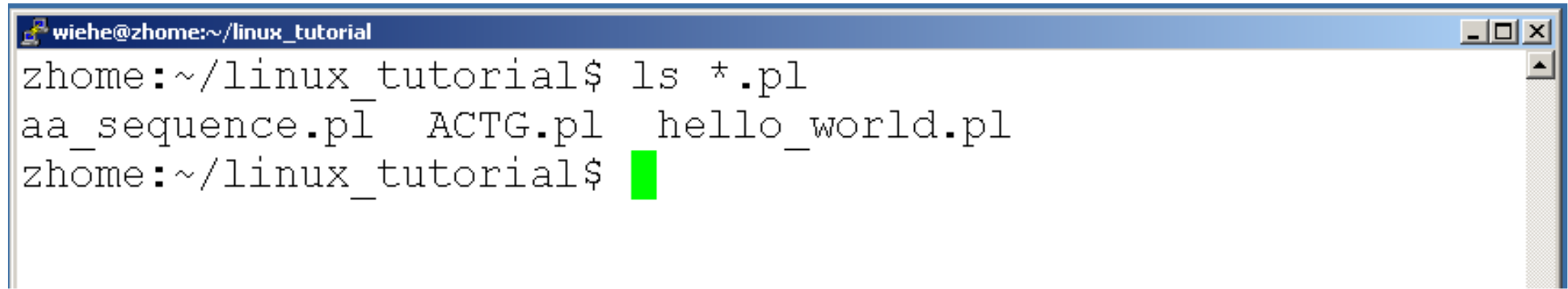
```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -ltr
total 20
-rw-rw-r-- 1 wiehe wiehe 92 Aug 30 11:54 ACTG.pl
-rw-rw-r-- 1 wiehe wiehe 169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 24 Aug 30 12:23 output.txt
-rw-rw-r-- 1 wiehe wiehe 21 Aug 30 12:23 data.dat
zhome:~/linux_tutorial$
```



Linux Terminal Commands

General Syntax: *

“*” can be used as a wildcard in unix/linux



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls *.pl
aa_sequence.pl  ACTG.pl  hello_world.pl
zhome:~/linux_tutorial$
```



- **cd (change [current working] directory)**

To change to a specific directory use “cd”

```
$ cd path
```

```
$ cd ../.. (Enter) Output    root
```

```
$ cd ..    (Enter) Output    one step backward (Parent directory)
```

```
$ cd       (Enter) Output    home directory
```

Or the path to a particular distention

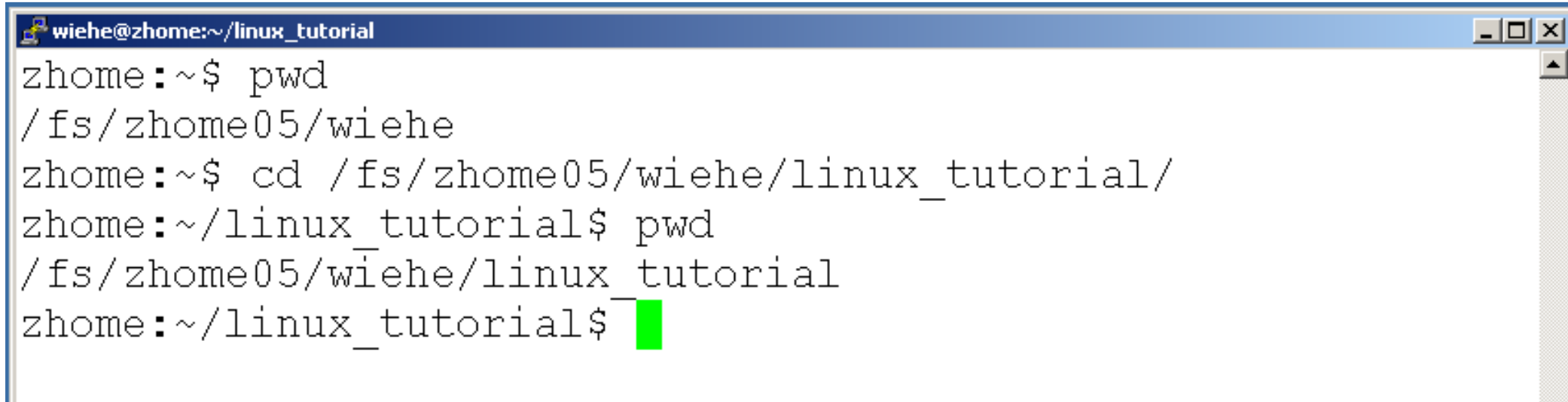
(the current home dir /directory name

/directory name from root ../user or specific directory/.....



- **cd (change [current working] directory)**

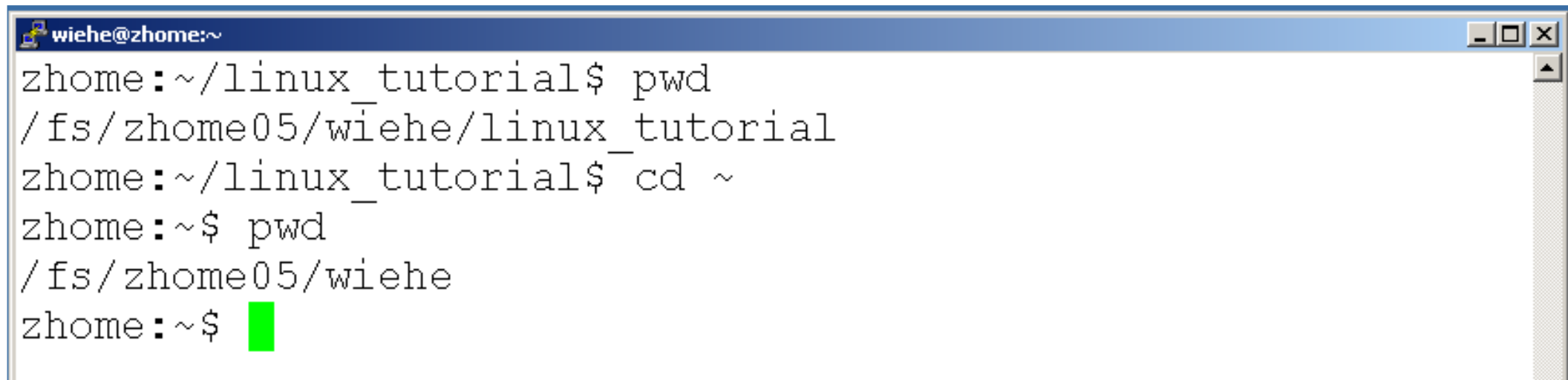
To change to a specific directory use “cd”

A screenshot of a Linux terminal window. The title bar shows the user 'wiehe' at host 'zhome' in the directory '~/linux_tutorial'. The terminal content shows a sequence of commands: 'pwd' returns '/fs/zhome05/wiehe', 'cd /fs/zhome05/wiehe/linux_tutorial/' changes the directory, and a second 'pwd' returns '/fs/zhome05/wiehe/linux_tutorial'. The prompt is now 'zhome:~/linux_tutorial\$' with a green cursor.

```
wiehe@zhome:~/linux_tutorial
zhome:~$ pwd
/fs/zhome05/wiehe
zhome:~$ cd /fs/zhome05/wiehe/linux_tutorial/
zhome:~/linux_tutorial$ pwd
/fs/zhome05/wiehe/linux_tutorial
zhome:~/linux_tutorial$
```

Linux Terminal Commands

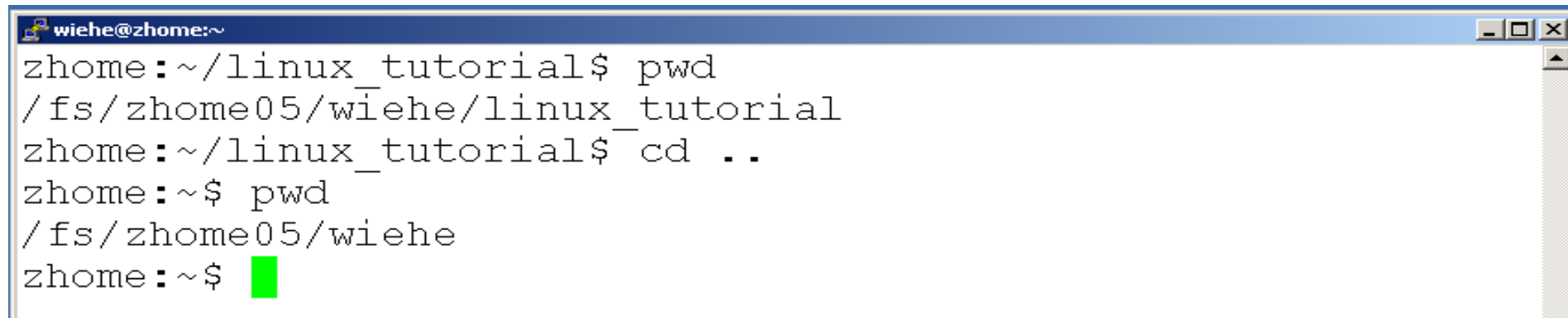
- **cd (change [current working] directory)**
“~” is the location of your home directory”



```
wiehe@zhome:~  
zhome:~/linux_tutorial$ pwd  
/fs/zhome05/wiehe/linux_tutorial  
zhome:~/linux_tutorial$ cd ~  
zhome:~$ pwd  
/fs/zhome05/wiehe  
zhome:~$
```



- **cd (change [current working] directory)**
 - “..” is the location of the directory below current one

A screenshot of a Linux terminal window. The title bar shows 'wiehe@zhome:~'. The terminal content shows a sequence of commands and their outputs: 'pwd' returns '/fs/zhome05/wiehe/linux_tutorial', 'cd ..' returns to the home directory, and another 'pwd' confirms the path '/fs/zhome05/wiehe'. A green cursor is visible at the end of the final prompt.

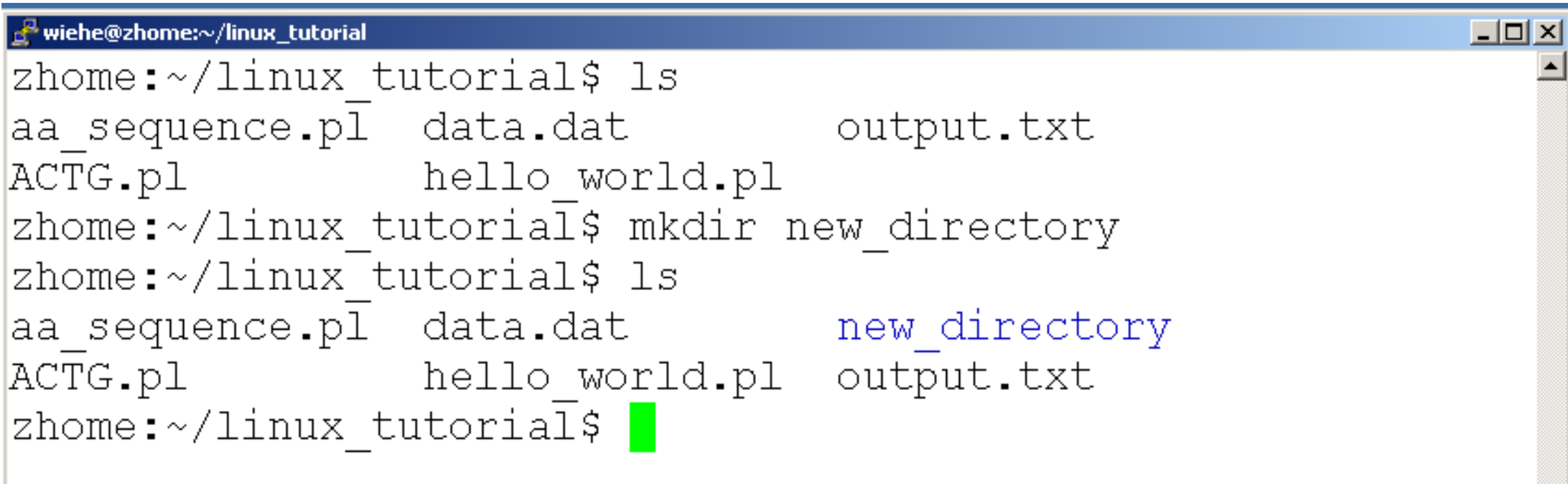
```
wiehe@zhome:~  
zhome:~/linux_tutorial$ pwd  
/fs/zhome05/wiehe/linux_tutorial  
zhome:~/linux_tutorial$ cd ..  
zhome:~$ pwd  
/fs/zhome05/wiehe  
zhome:~$
```



- **mkdir (make directory)**

To create a new directory use “mkdir”

\$ mkdir OS2016/17 output (OS2016/17 Subdirectory in current directory)

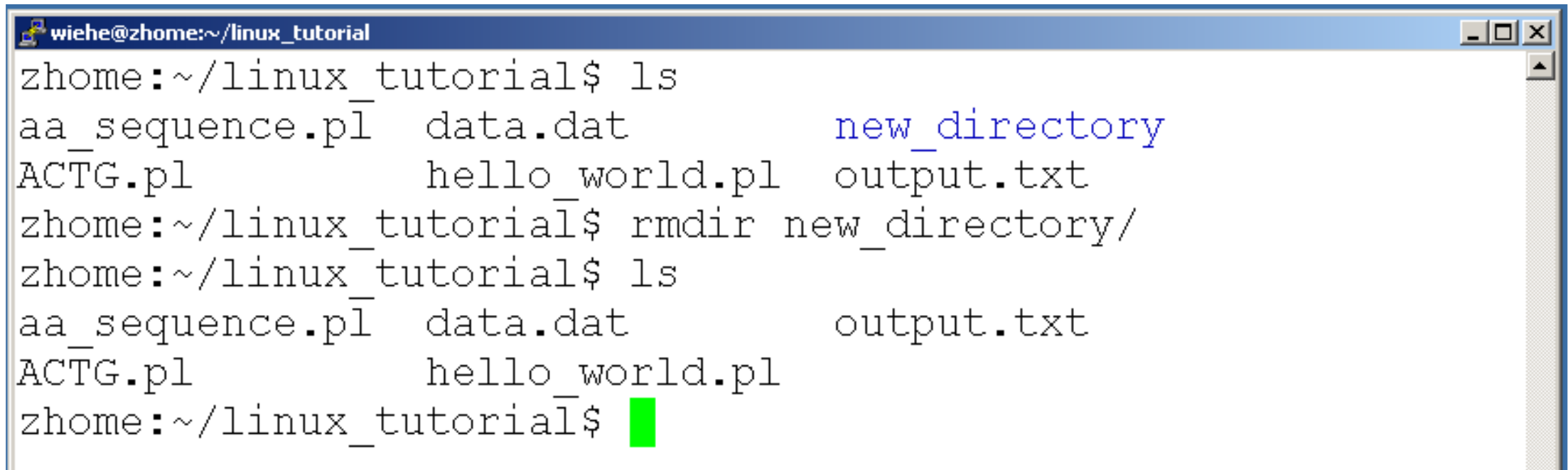
A screenshot of a Linux terminal window. The title bar shows the user 'wiehe' at host 'zhome' in the directory '~/linux_tutorial'. The terminal content shows the following sequence of commands and output:
1. Command: `ls`
Output: `aa_sequence.pl data.dat output.txt`
2. Command: `ACTG.pl hello_world.pl`
3. Command: `mkdir new_directory`
4. Command: `ls`
Output: `aa_sequence.pl data.dat new_directory`
5. Command: `ACTG.pl hello_world.pl output.txt`
6. The prompt `zhome:~/linux_tutorial$` is followed by a green cursor.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat          output.txt
ACTG.pl        hello_world.pl
zhome:~/linux_tutorial$ mkdir new_directory
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat          new_directory
ACTG.pl        hello_world.pl  output.txt
zhome:~/linux_tutorial$ █
```

- **rmdir (remove directory)**

To remove an empty directory use “rmdir”

\$ rmdir OS2016/17 output (Removes Subdirectory OS2016/17 in current directory)



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat          new_directory
ACTG.pl         hello_world.pl    output.txt
zhome:~/linux_tutorial$ rmdir new_directory/
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat          output.txt
ACTG.pl         hello_world.pl
zhome:~/linux_tutorial$
```



Creating files in Unix/Linux

Various Editors:

- 1) gedit
- 2) nano / pico
- 3) vi
- 4) emacs



Editing a file using pico or nano

Type “pico” or “nano” at the prompt



The screenshot shows the nano text editor running in a terminal window. The title bar indicates the user is 'wiehe' at 'zhome' in the directory '~/linux_tutorial'. The editor's status bar at the top shows 'UW PICO(tm) 4.6' and 'New Buffer'. The main editing area is empty, with a green cursor block on the first line. On the right side, there is a small box labeled 'Learning UNIX'. The bottom of the screen displays a list of keyboard shortcuts for navigation and editing, such as '^G Get Help', '^O Write Out', '^R Read From File', '^Y Prev Page', '^K Cut Text', '^C Cur Pos', '^X Exit', '^J Justify', '^W Where', '^V Next Page', '^U UnCut', '^T To Search', and '^_ Undo'.

```
wiehe@zhome:~/linux_tutorial
UW PICO(tm) 4.6      New Buffer

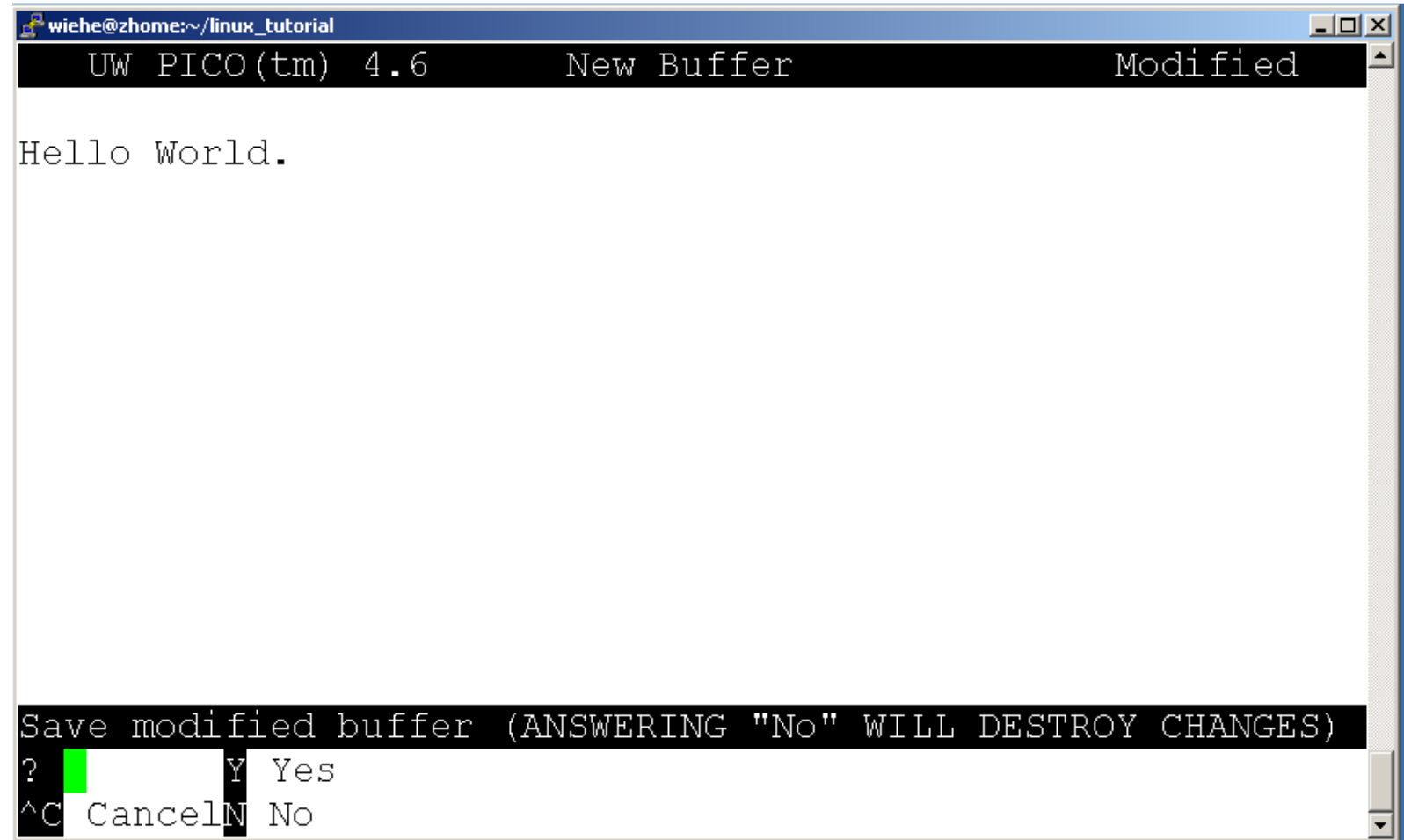
Learning UNIX

^G Get Help ^O Write Out ^R Read From File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit     ^J Justify  ^W Where    ^V Next Page ^U UnCut    ^T To Search ^_ Undo
```



Editing a file using pico or nano

To save use “ctrl-x”



```
wiehe@zhome:~/linux_tutorial
UW PICO(tm) 4.6      New Buffer      Modified

Hello World.

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES)
? ☐ Yes
^C ☐ No
```



Displaying a file

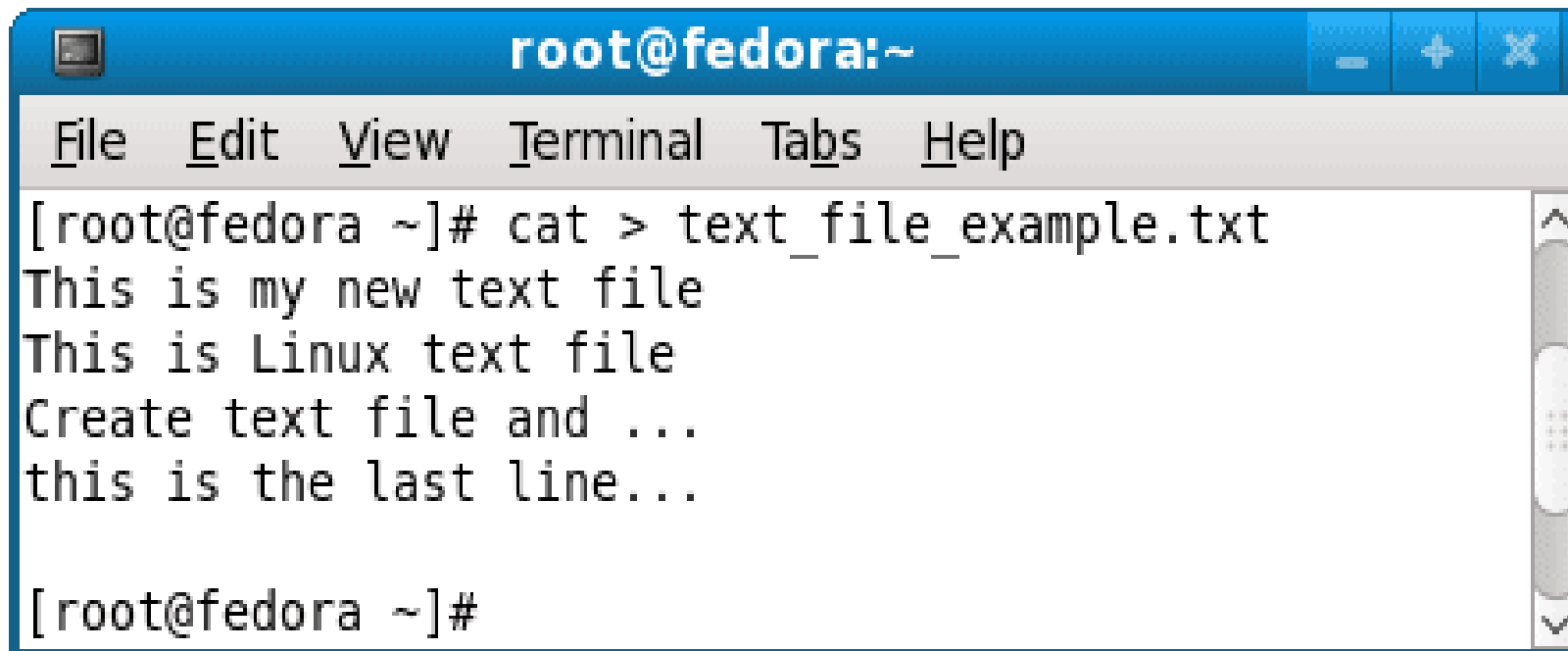
Various ways to display a file in Unix

- cat
- less
- head
- tail



Command: cat

- Dumps an entire file to standard output
- Good for displaying short, simple files

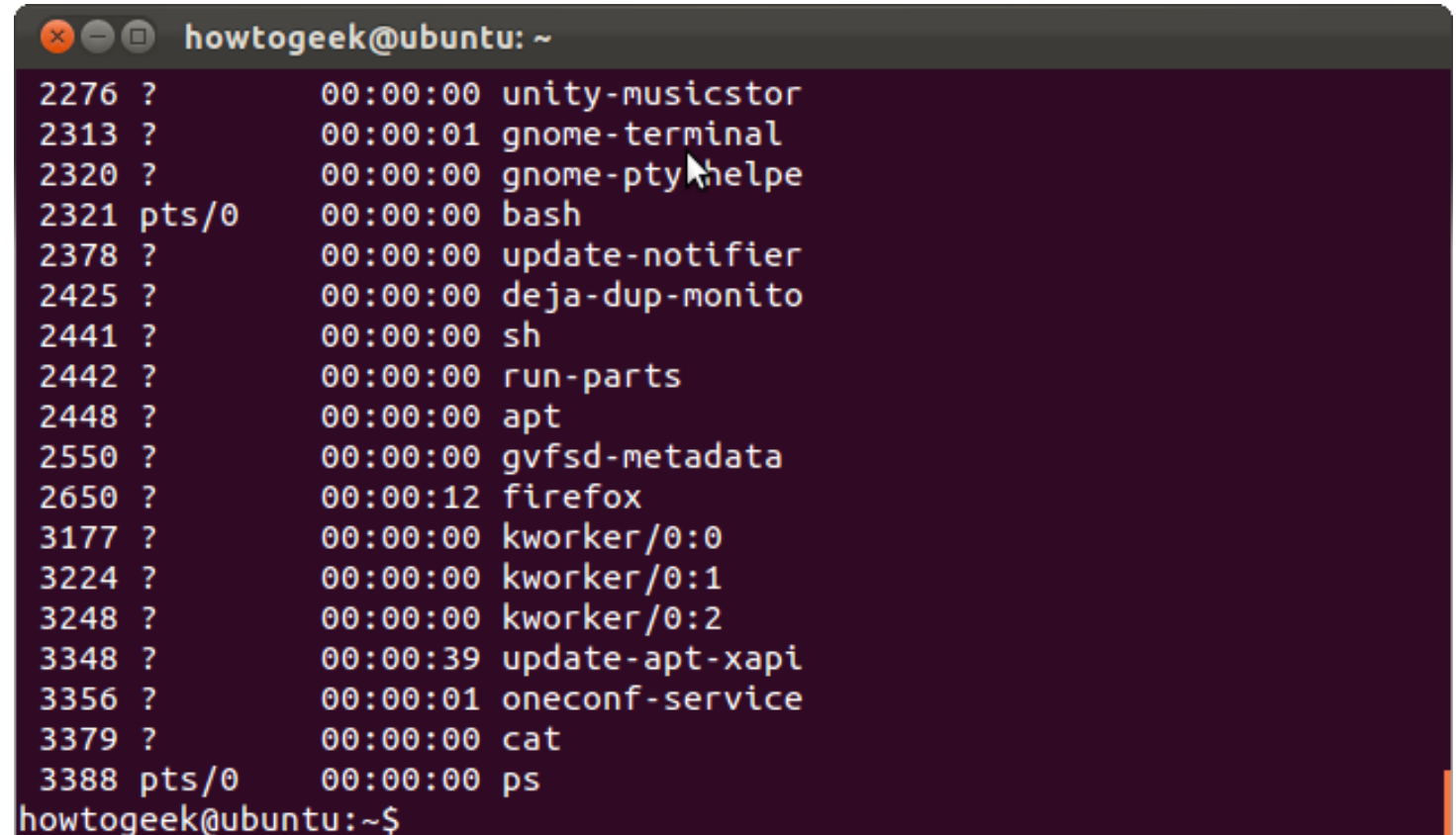


```
root@fedora:~  
File Edit View Terminal Tabs Help  
[root@fedora ~]# cat > text_file_example.txt  
This is my new text file  
This is Linux text file  
Create text file and ...  
this is the last line...  
[root@fedora ~]#
```



Command: less

- less” displays a file, allowing forward/backward movement within it
 - return scrolls forward one line, space one page
 - y scrolls back one line, b one page
- use “/” to search for a string
- Press q to quit



```
howtogeek@ubuntu: ~  
2276 ?      00:00:00 unity-musicstor  
2313 ?      00:00:01 gnome-terminal  
2320 ?      00:00:00 gnome-ptyhelpe  
2321 pts/0   00:00:00 bash  
2378 ?      00:00:00 update-notifier  
2425 ?      00:00:00 deja-dup-monito  
2441 ?      00:00:00 sh  
2442 ?      00:00:00 run-parts  
2448 ?      00:00:00 apt  
2550 ?      00:00:00 gvfsd-metadata  
2650 ?      00:00:12 firefox  
3177 ?      00:00:00 kworker/0:0  
3224 ?      00:00:00 kworker/0:1  
3248 ?      00:00:00 kworker/0:2  
3348 ?      00:00:39 update-apt-xapi  
3356 ?      00:00:01 oneconf-service  
3379 ?      00:00:00 cat  
3388 pts/0   00:00:00 ps  
howtogeek@ubuntu:~$
```



Command: head

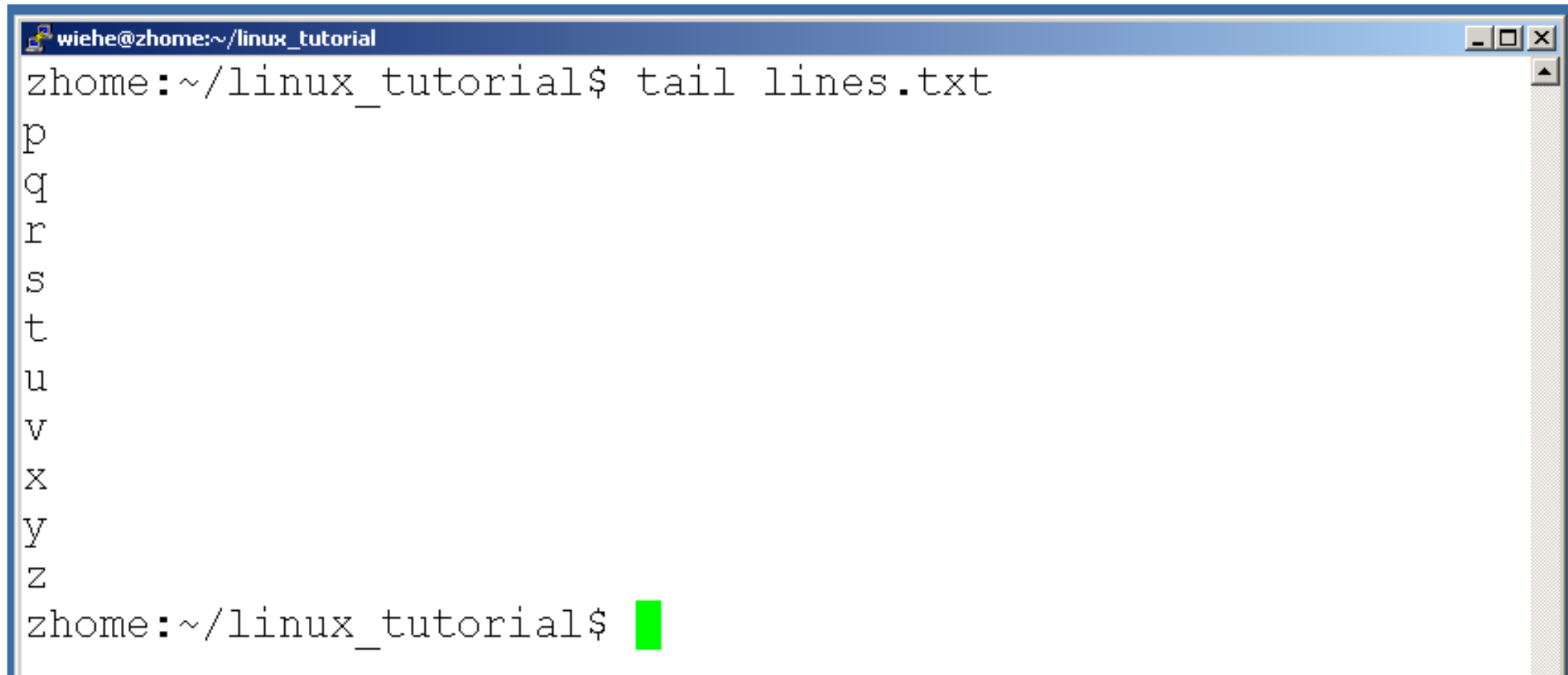
- “head” displays the top part of a file
- By default it shows the first 10 lines
- -n option allows you to change that
- “head -n50 file.txt” displays the first 50 lines of file.txt



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ head lines.txt
a
b
c
d
e
f
g
h
i
j
zhome:~/linux_tutorial$
```

Command: tail

Same as head, but shows the last lines

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The terminal shows the command 'tail lines.txt' being executed. The output consists of the letters 'p', 'q', 'r', 's', 't', 'u', 'v', 'x', 'y', and 'z' on separate lines. The prompt 'zhome:~/linux_tutorial\$' is followed by a green cursor block.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ tail lines.txt
p
q
r
s
t
u
v
x
y
z
zhome:~/linux_tutorial$
```

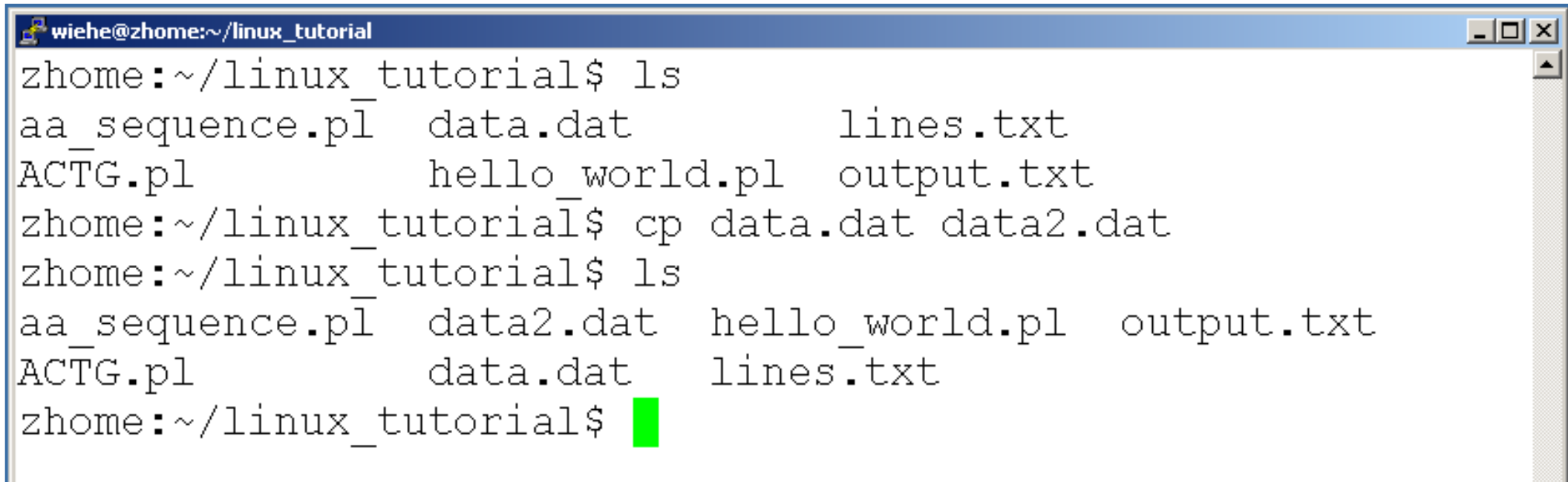


- **cp (copy)** To copy a file use “cp”

\$ cp source-file(s) destination

To copy entire directories (including their contents), use a recursive copy:

\$ cp -rd source-directories destination-directory

A terminal window titled 'wiehe@zhome:~/linux_tutorial' with standard window controls. It shows a sequence of commands and their outputs. First, 'ls' lists files: aa_sequence.pl, data.dat, lines.txt, ACTG.pl, hello_world.pl, and output.txt. Then, 'cp data.dat data2.dat' is executed. A second 'ls' command shows the updated directory contents, where data.dat has been replaced by data2.dat.

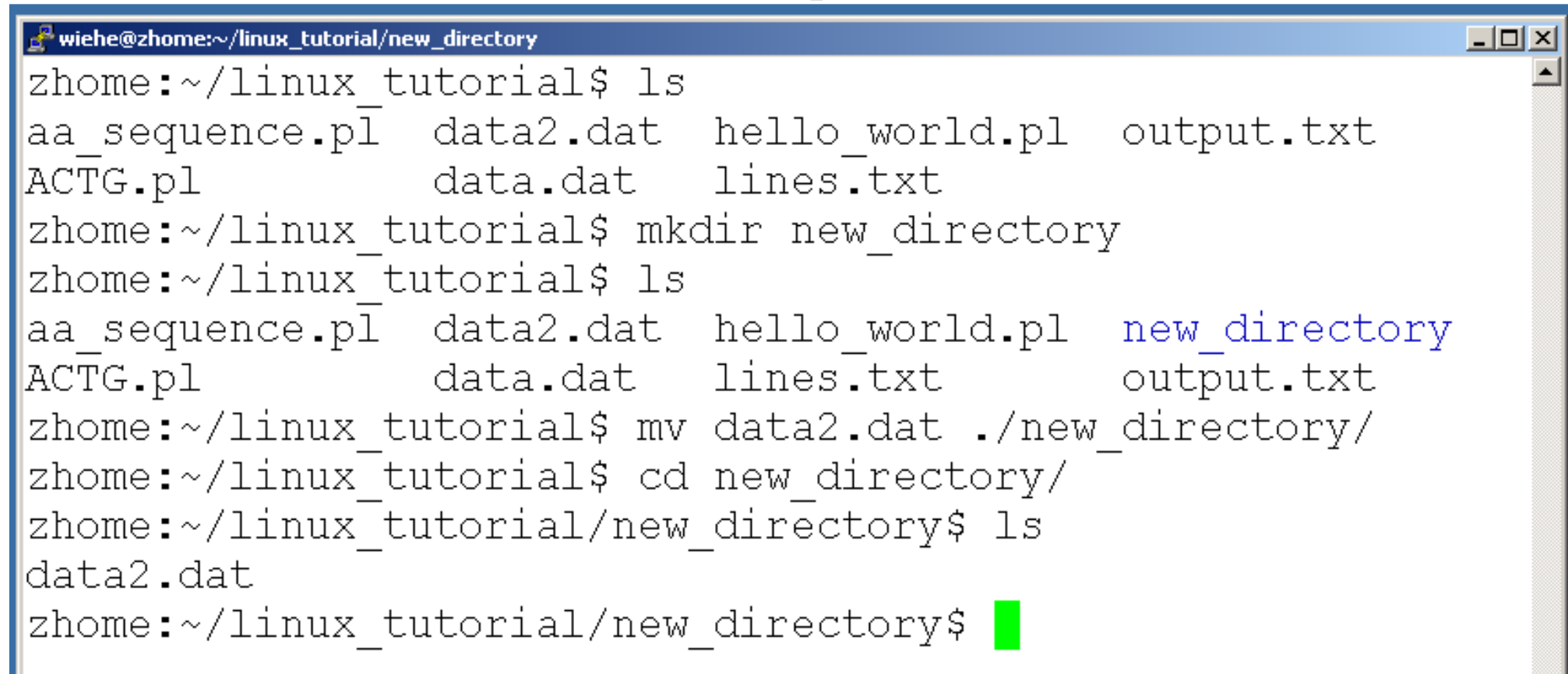
```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      lines.txt
ACTG.pl        hello_world.pl output.txt
zhome:~/linux_tutorial$ cp data.dat data2.dat
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data2.dat    hello_world.pl  output.txt
ACTG.pl        data.dat     lines.txt
zhome:~/linux_tutorial$
```



- **mv (move/rename)**

\$ mv source destination

is used to rename files/directories and/or move them from one directory into another. Exactly one source and one destination must be specified.

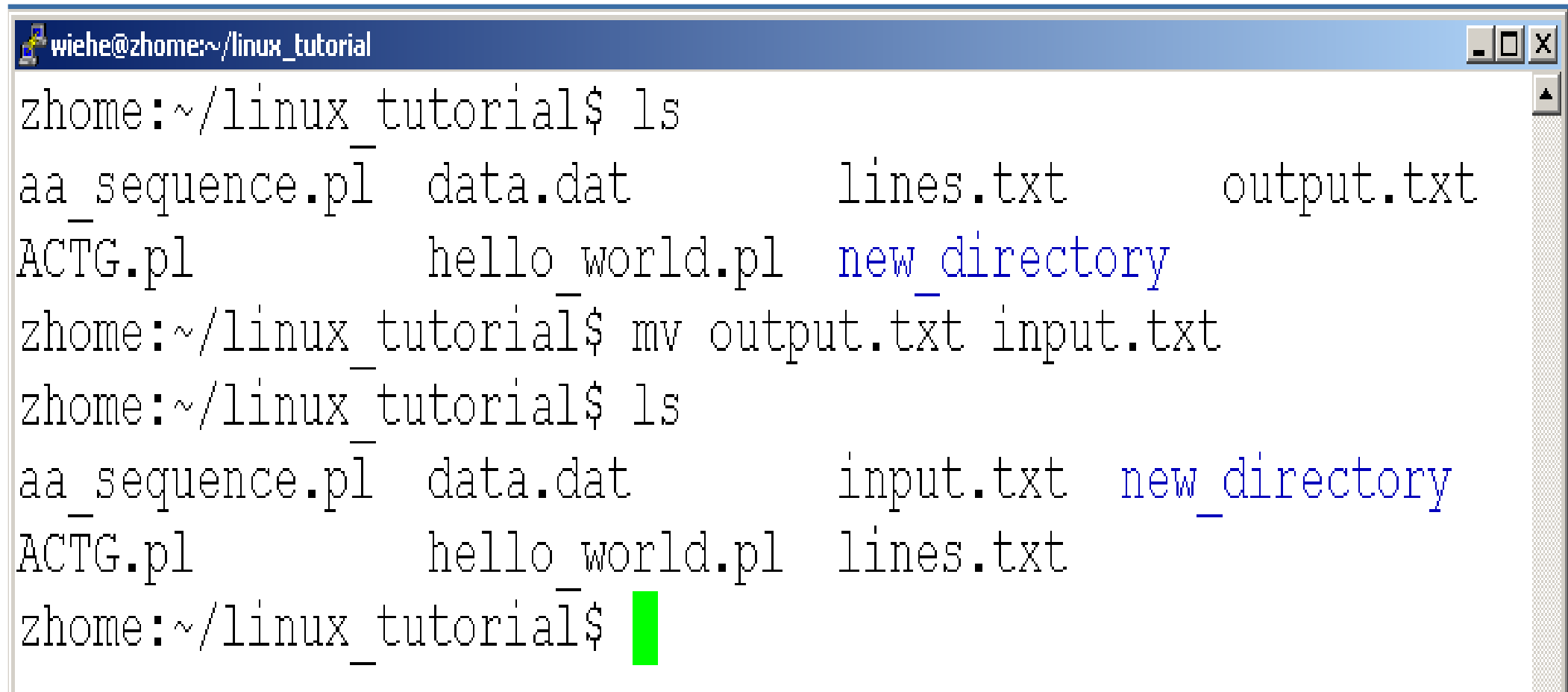


```
wiehe@zhome:~/linux_tutorial/new_directory
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data2.dat  hello_world.pl  output.txt
ACTG.pl        data.dat   lines.txt
zhome:~/linux_tutorial$ mkdir new_directory
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data2.dat  hello_world.pl  new_directory
ACTG.pl        data.dat   lines.txt       output.txt
zhome:~/linux_tutorial$ mv data2.dat ./new_directory/
zhome:~/linux_tutorial$ cd new_directory/
zhome:~/linux_tutorial/new_directory$ ls
data2.dat
zhome:~/linux_tutorial/new_directory$
```



- **mv (rename)**

\$ mv source destination



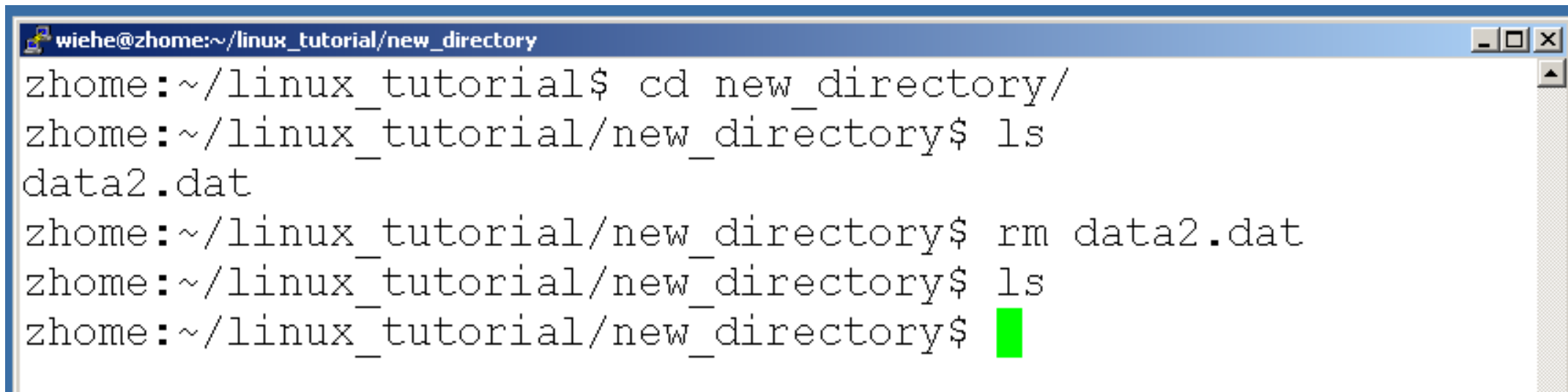
```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      lines.txt      output.txt
ACTG.pl         hello_world.pl new_directory
zhome:~/linux_tutorial$ mv output.txt input.txt
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      input.txt      new_directory
ACTG.pl         hello_world.pl lines.txt
zhome:~/linux_tutorial$
```

- **rm (remove/delete)**

To remove a file use “rm”

\$ rm target-file(s) (works as shift delete)

-r (recursive) -f (force) -rf (forces deleting everything)

A screenshot of a Linux terminal window. The title bar shows the user 'wiehe' at 'zhome' in the directory '~/linux_tutorial/new_directory'. The terminal content shows a sequence of commands: 'cd new_directory/' to change to the new directory, 'ls' to list files (showing 'data2.dat'), 'rm data2.dat' to remove the file, and another 'ls' command which results in a blank line, indicating the file has been successfully removed. A green cursor is visible at the end of the last command line.

```
wiehe@zhome:~/linux_tutorial/new_directory
zhome:~/linux_tutorial$ cd new_directory/
zhome:~/linux_tutorial/new_directory$ ls
data2.dat
zhome:~/linux_tutorial/new_directory$ rm data2.dat
zhome:~/linux_tutorial/new_directory$ ls
zhome:~/linux_tutorial/new_directory$
```



- **rm (remove/delete)**

\$ rm target-file(s) (works as shift delete)

-r (recursive) -f (force) -rf (forces deleting everything)

To remove a file “recursively”: rm -r

Used to remove all files and directories

Be very careful, deletions are permanent in Unix/Linux

- **Sudo Superuser**



“r” means “read only” permission

“w” means “write” permission

“x” means “execute” permission

In case of directory, “x” grants permission to list directory contents



File and Directory Permissions

<u>Permission</u>	<u>File</u>	<u>Directory</u>
read	User can look at the contents of the file	User can list the files in the directory
write	User can modify the contents of the file	User can create new files and remove existing files in the directory
execute	User can use the filename as a UNIX command	User can change into the directory, but cannot list the files unless (s)he has read permission. User can read files if (s)he has read permission on them.



File and Directory Permissions

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r-- 1 wiehe wiehe 169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r-- 1 wiehe wiehe 92 Aug 30 11:54 ACTG.pl
-rw-rw-r-- 1 wiehe wiehe 21 Aug 30 12:23 data.dat
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 24 Aug 30 12:23 input.txt
-rw-rw-r-- 1 wiehe wiehe 50 Aug 30 13:13 lines.txt
drwxrwxr-x 2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```



File and Directory Permissions

51

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r-- 1 wiehe wiehe 169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r-- 1 wiehe wiehe 92 Aug 30 11:54 ACTG.pl
-rw-rw-r-- 1 wiehe wiehe 21 Aug 30 12:23 data.dat
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 24 Aug 30 12:23 input.txt
-rw-rw-r-- 1 wiehe wiehe 50 Aug 30 13:13 lines.txt
drwxrwxr-x 2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```

“The World”

Group

User (you)



File and Directory Permissions

---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
rw-	6
rwX	7



File and Directory Permissions

chmod (change [file or directory] mode)

\$ chmod options files

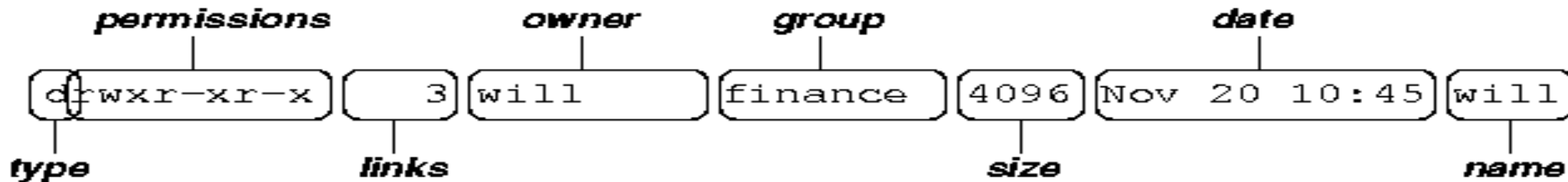
If you own the file, you can change it's permissions with “chmod”

Syntax: chmod [**u**ser/**g**roup/**o**thers/**a**ll]+[permission] [file(s)]

For example the command:

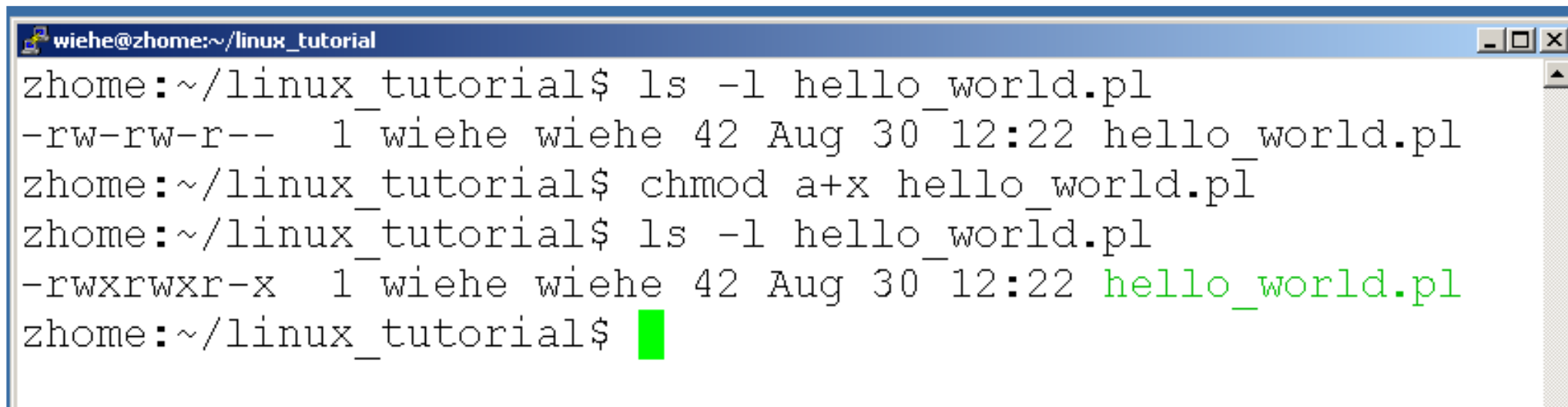
\$ chmod 600 private.txt

\$ chmod ug=rw,o-rw,a-x *.txt



File and Directory Permissions

Below we grant execute permission to all:



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -l hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
zhome:~/linux_tutorial$ chmod a+x hello_world.pl
zhome:~/linux_tutorial$ ls -l hello_world.pl
-rwxrwxr-x 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
zhome:~/linux_tutorial$
```



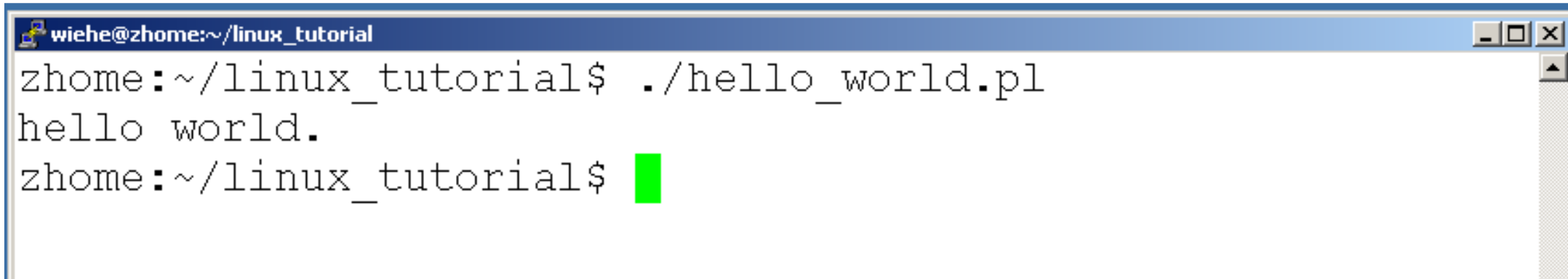
Running a program (a.k.a. a job)

Make sure the program has executable permissions

Use “./” to run the program

e.g:

Running the sample perl script “hello_world.pl”

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The terminal shows the command './hello_world.pl' being executed, followed by the output 'hello world.' and a new prompt line 'zhome:~/linux_tutorial\$' with a green cursor.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ./hello_world.pl
hello world.
zhome:~/linux_tutorial$
```



Ending a program

To end a program use “ctrl-c”. To try it:



Command: ps

To view the processes that you're running:



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1255 pts/2        00:00:01 ACTG.pl
 1270 pts/2        00:00:00 ps
zhome:~/linux_tutorial$
```

Command: top

To view the CPU usage of all processes:

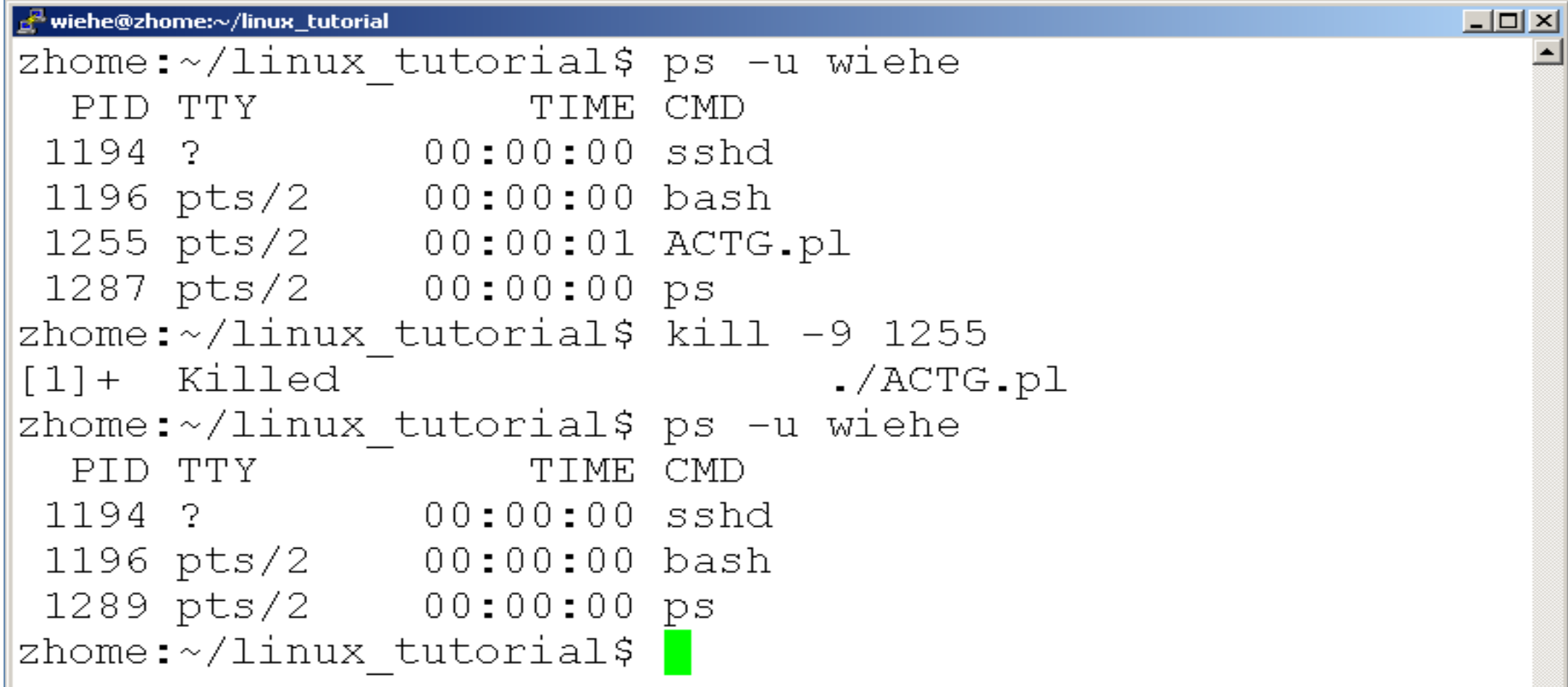
```
wiehe@zhome:~/linux_tutorial
top - 13:46:33 up 50 days,  4:26,  2 users,  load avera
Tasks:  total,      running,      sleeping,      stoppe
Cpu(s) :    us,      sy,          ni,          id,          w
Mem:      total,          used,          free,
Swap:      total,          used,          free,

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM
3403	root	15	0	0	0	0	S	0.7	0.0
1	root	16	0	1604	324	292	S	0.0	0.0
2	root	RT	0	0	0	0	S	0.0	0.0
3	root	34	19	0	0	0	S	0.0	0.0
4	root	RT	0	0	0	0	S	0.0	0.0
5	root	34	19	0	0	0	S	0.0	0.0
6	root	RT	0	0	0	0	S	0.0	0.0
7	root	34	19	0	0	0	S	0.0	0.0
8	root	RT	0	0	0	0	S	0.0	0.0
9	root	34	19	0	0	0	S	0.0	0.0

Command: kill

To terminate a process use “kill”



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1255 pts/2        00:00:01 ACTG.pl
 1287 pts/2        00:00:00 ps
zhome:~/linux_tutorial$ kill -9 1255
[1]+  Killed                  ./ACTG.pl
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1289 pts/2        00:00:00 ps
zhome:~/linux_tutorial$
```



Input/Output Redirection (“piping”)

- Programs can output to other programs
- Called “piping”
- “program_a | program_b”
 - program_a’s output becomes program_b’s input
- “program_a > file.txt”
 - program_a’s output is written to a file called “file.txt”
- “program_a < input.txt”
 - program_a gets its input from a file called “input.txt”



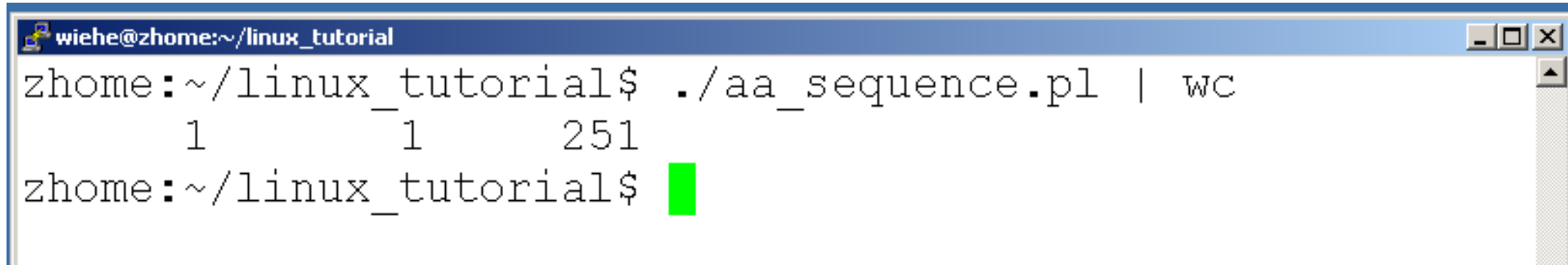
A few examples of piping

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  hello_world.pl  new_directory
ACTG.pl        input.txt
data.dat       lines.txt
zhome:~/linux_tutorial$ ./aa_sequence.pl > sequence.txt
zhome:~/linux_tutorial$ ls
aa_sequence.pl  hello_world.pl  new_directory
ACTG.pl        input.txt       sequence.txt
data.dat       lines.txt
zhome:~/linux_tutorial$ less sequence.txt
```



Command: wc

- To count the characters, words, and lines in a file use “wc”
- The first column in the output is lines, the second is words, and the last is characters

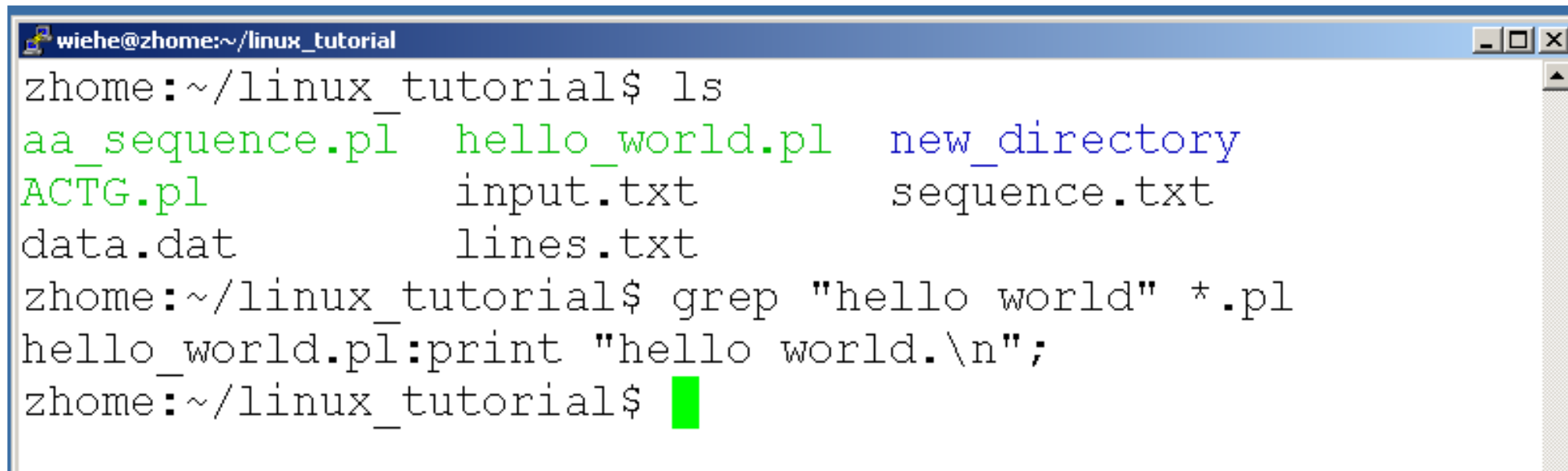


```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ./aa_sequence.pl | wc
      1      1     251
zhome:~/linux_tutorial$
```



Command: grep

To search files in a directory for a specific string use “grep”



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  hello_world.pl  new_directory
ACTG.pl         input.txt       sequence.txt
data.dat        lines.txt
zhome:~/linux_tutorial$ grep "hello world" *.pl
hello_world.pl:print "hello world.\n";
zhome:~/linux_tutorial$
```



Command: diff

To compare to files for differences use “diff”

Try: `diff /dev/null hello.txt`

`/dev/null` is a special address -- it is always empty, and anything moved there is deleted



ssh, scp

ssh is used to securely log in to remote systems, successor to telnet

`ssh [username]@[hostname]`

Try:

`ssh yourusername@localhost`

Type “exit” to log out of session

Scp is used to copy files to/from remote systems, syntax is similar to cp:

`scp [local path] [username]@[hostname]:[remote file path]`

Try:

`scp hello.txt yourusername@localhost:scp-test.txt`



Youtube

https://www.youtube.com/watch?v=9t_gJWC32zk

Linux Handout & Tutorial

<http://www.guru99.com/unix-linux-tutorial.html>

William Knottenbelt Imperial college London 2001

<http://www.doc.ic.ac.uk/~wjk/UnixIntro/index.html>

WORLD OF ASIC 2014

<http://www.asic-world.com/scripting/unix3.html>



<http://www.ee.surrey.ac.uk/Teaching/Unix/>

<http://www.ugu.com/sui/ugu/show?help.beginners>

<http://en.wikipedia.org/wiki/Unix>

