

Getting Started

Logging In and Out



Getting started: logging in and out

- # Every user in UNIX has a **username** (also called **login name** or **account name**) and a **password**.
- # Logging in requires that you enter your username and password.
- # Your username and initial password are given to you by the system administrator.



Logging in from a terminal window (1)

- # You have to identify yourself by typing your username at the *login prompt*:

login:

and then your password at the *password prompt*:

login: stewart

Password:

- # When you type the password, you won't see the cursor move until you press the Enter key.



Logging in from a terminal window (2)

- # If you enter either your username or password incorrectly, you will see this message:

login incorrect

login:

and you will have to repeat the previous steps.

Username are case-sensitive!! You have to type them exactly as they are given to you.



Logging in from a terminal window (3)

- # If you are successful, you will see a message something like this:

```
Last login: Mon Jun 26 21:32:48  
$
```

The '\$' is called the *shell prompt*. It is the indication by UNIX that it is waiting for you to enter a command.



Changing your password (1)

- # When you login for the first time, you must change your password. No exceptions.
- # Type the command '**passwd**' at the prompt. Note that it is '**password**' without the '**or**'.
- # You will be asked to enter the current password and then asked for the new password, twice. If you make any mistakes, the password will not be changed.



Changing your password (2)

- # The session will look something like this:

```
$ passwd
Changing password for user sweiss.
(current) UNIX password: you type here
New UNIX password: you type here
Retype new UNIX password: you type here
passwd: password has been changed.
$
```



About passwords

- # Your password must be difficult to "crack". Some rules to follow:
 - Must be at least 8 characters long
 - Must contain a digit, an uppercase and lowercase letter
 - Must contain punctuation (period, comma, hyphen, etc)
 - Must not be a word in a dictionary or encyclopedia
 - Should not be guessable (your birthday, ss#, etc)



Logging out

- # Before going any further, try logging out.
- # To logout, type "**logout**" at the shell prompt. This is the safest way to logout:

```
$ logout
```

- # You can also type "**exit**". They work the same in a *login shell*. (Soon you will learn what a login shell is.)



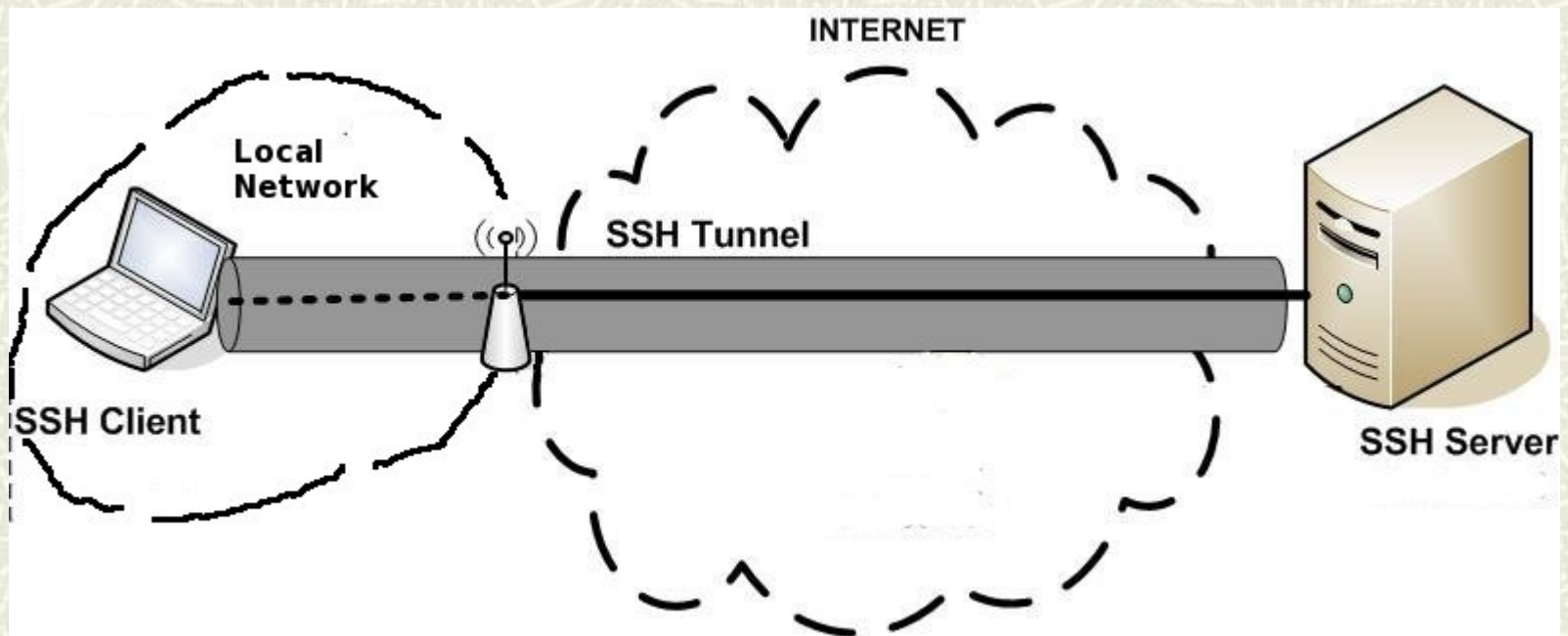
Logging in using SSH

- # To login to a UNIX machine over a network, you usually have to use **SSH**.
- # **SSH** is a protocol that encrypts what you type on your "local" computer, sends the encrypted text to the "remote" computer, and decrypts it there.
- # The remote computer has a program called an **SSH server**. Your computer has an **SSH client**.
- # SSH stands for "**secure shell**."



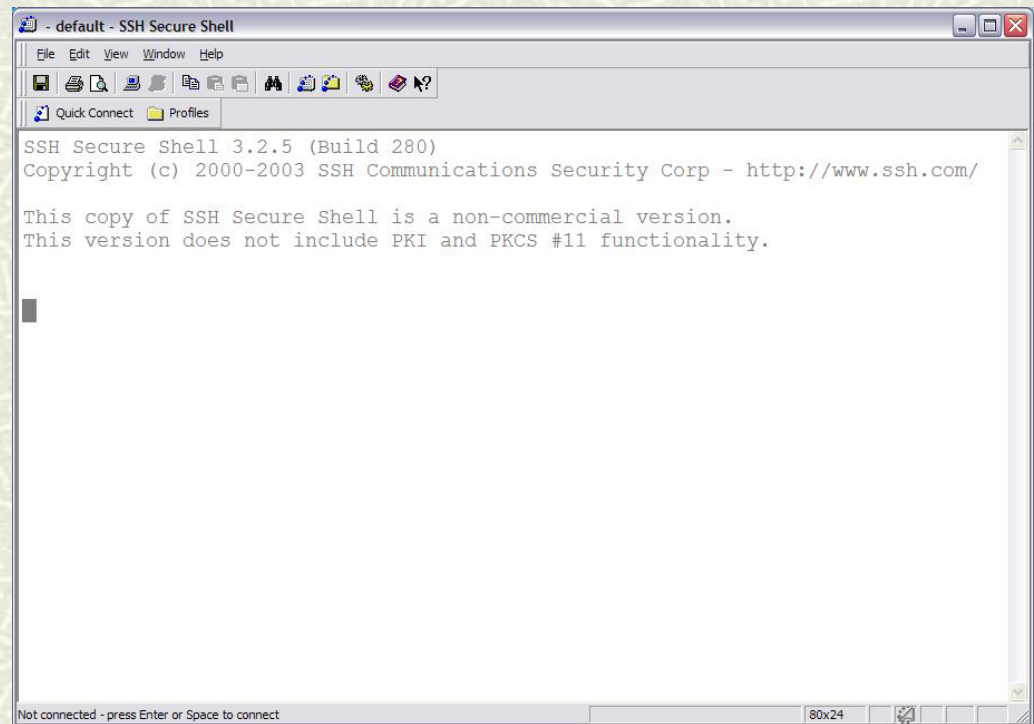
The SSH connection

Conceptually it looks like this::



SSH: step 1

You start up your SSH client and see a window something like the one on the right (which is what you see using the Windows SSH client.)



SSH: step 2

Find the button that connects to a server. It is usually called **Connect** or something similar. You must then enter the *full name* of the host (e.g. **eniac.cs.hunter.cuny.edu**) and your *login* name.



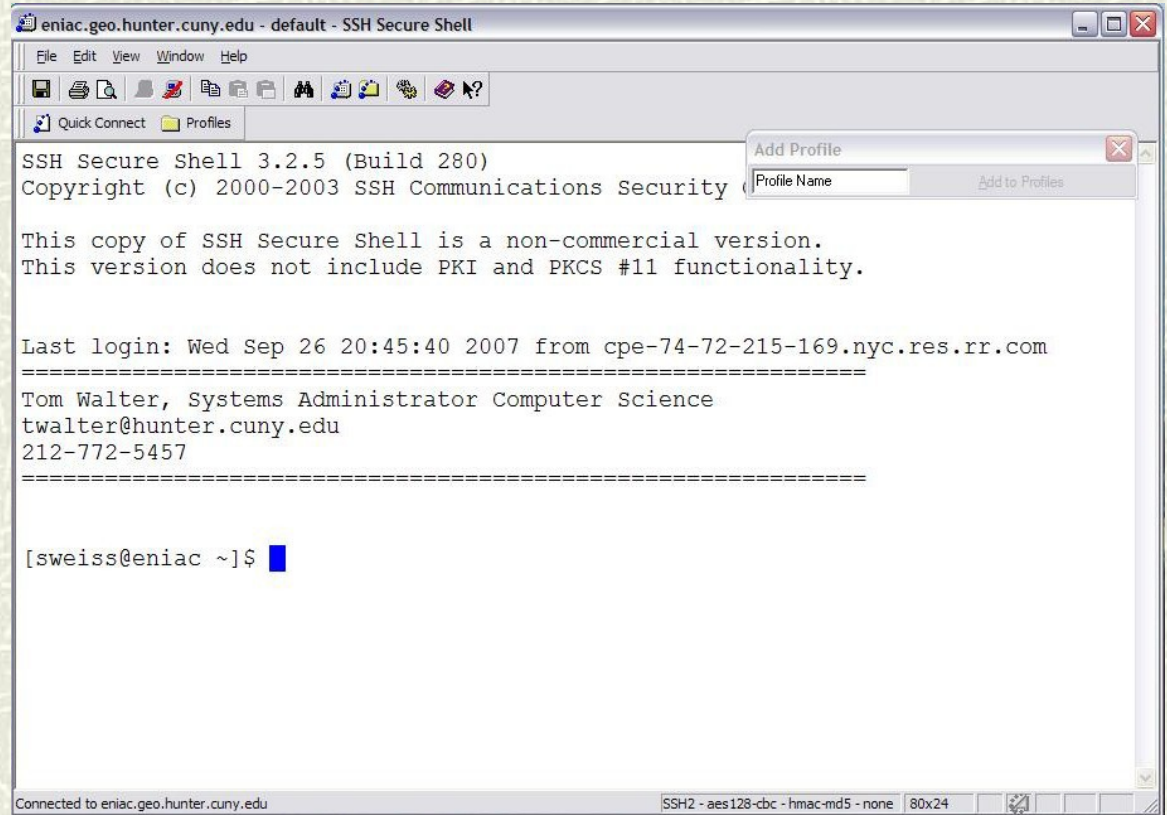
SSH: step 3

If you enter the correct hostname and login name, you will then see a window in which you can enter your password. It might look like the one below.



SSH: step 4

If the password is correct, you will then be at the command prompt, as shown on the right.



The screenshot shows a terminal window titled "eniaco.geo.hunter.cuny.edu - default - SSH Secure Shell". The window contains the following text:

```
SSH Secure Shell 3.2.5 (Build 280)
Copyright (c) 2000-2003 SSH Communications Security

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.

Last login: Wed Sep 26 20:45:40 2007 from cpe-74-72-215-169.nyc.res.rr.com
=====
Tom Walter, Systems Administrator Computer Science
twalter@hunter.cuny.edu
212-772-5457
=====

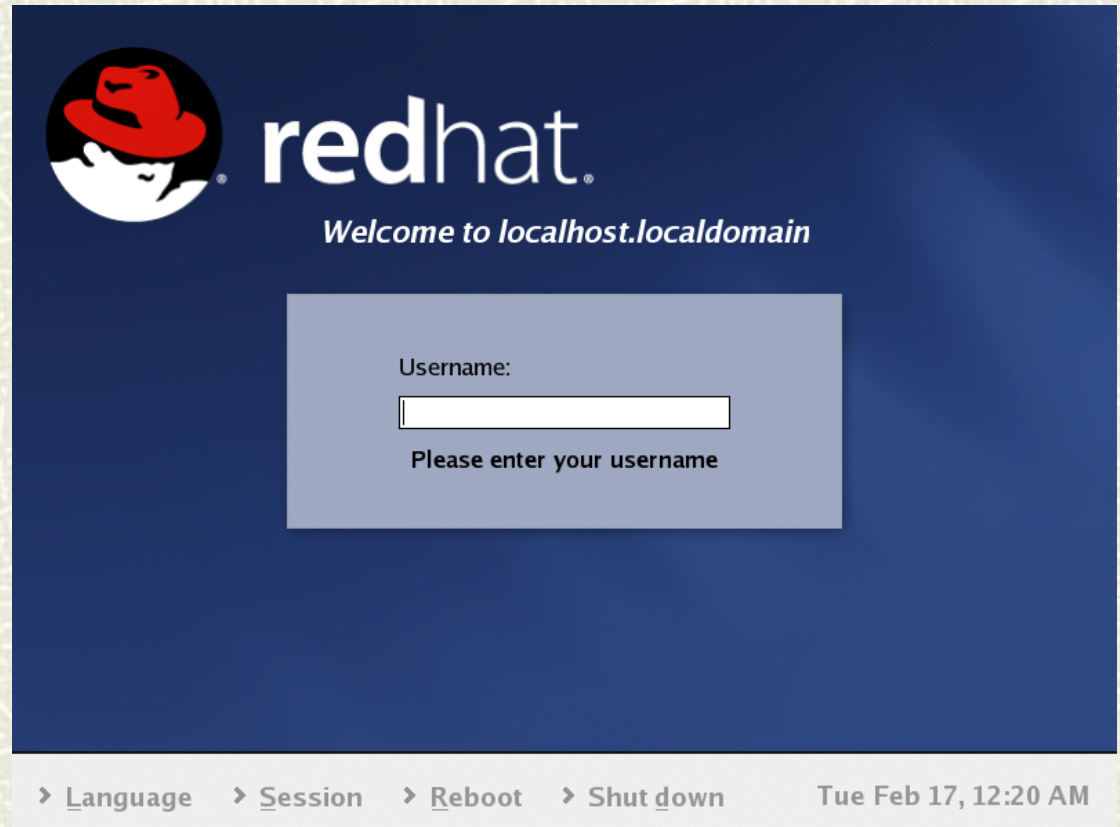
[sweiss@eniaco ~]$
```

The terminal window also shows a status bar at the bottom indicating "Connected to eniac.geo.hunter.cuny.edu" and "SSH2 - aes128-cbc - hmac-md5 - none 80x24".



Logging in at the console

If you are sitting at a Unix machine, the login procedure will be slightly different. You will see a screen something like this one. This is the old RedHat screen. The Ubuntu screen looks different.



Logging in at the console (2)

- # You just enter your login name, after which you will be prompted to enter your password.
- # When you sit at the machine, the monitor is called a *console*.
- # After you login, you will have a desktop environment like the one found in Windows and Macintosh computers.



Opening a “Terminal” Window

- # If you are sitting at the console, to open a terminal window, you need to start a terminal application.
- # On most versions of Linux, there will be an *Applications* menu, and within that menu, a sub-menu named *System Tools*. The System Tools menu usually has a *Terminal* application.
- # Find it and start it up. Then you can continue with the next slide.



Getting help in UNIX

- # UNIX provides a complete and extensive set of on-line documentation to help you find and understand the commands that you need. You only need to remember a three-letter command: **man**, which is short for "*manual*."
- # The *man pages* are a collection of individual pages that describe all of the commands, system files, kernel functions, and library functions on your UNIX system.



The man command

- # Within the terminal, the **man** command displays the manual page that you specify. For example,
man passwd
will display the **passwd** man page, which describes how to use the **passwd** command.
man man
displays the **man** page for the **man** command itself, and is the best place to start learning about UNIX.



Ambiguity

- # Often, the same name is used for many purposes!
- # For example, **passwd** is a shell command, but **passwd** is also the name of the file that contains user login data. Typing
man passwd
displays the man page for the **passwd** shell command.
- # How can you see the man page for the **passwd** file?



Organization of manual pages

- # The manual pages are arranged in at least 9 *sections*. These typically include
 - User Commands
 - Unix System Calls
 - Library Functions
 - I/O and Special Files
 - File Formats and Miscellaneous
 - Games and Demos
 - Miscellaneous "useful information"
 - System Administrative Commands



Discovering the section numbers

The man page for **man** itself displays the section numbers of the various sections. In Fedora Linux, the first five sections are

- 1 User Commands
- 2 System Calls
- 3 C Library Functions
- 4 Devices and Special Files
- 5 File Formats and Conventions



More about man page organization

- # Different systems have these sections in different orders. You can type

man *sectnum* intro

where ***sectnum*** is a number like 1, 2, 3, etc. This will display the **intro** page for that manual section. The **intro** page will also tell you what pages are stored in that section. On my system,

man 5 intro

shows that Section 5 is "Files and File Formats".



Finding the man page for the passwd file

- Knowing that Section 5 contains man pages about files, I can type

```
man 5 passwd
```

and this will give me the man page for the **passwd** file. (If you are wondering why we didn't need the section number to get the man page for the **passwd command**, it is because the sections are searched in a specific order; if we omit the section number, we get the man page that is found first, usually in section 1.)



How to use a man page

Each man page consists of several parts, including:

- NAME
- SYNOPSIS
- DESCRIPTION
- OPTIONS
- SEE ALSO

and several others. **SYNOPSIS** shows you how to use the command. **DESCRIPTION** is a summary. **SEE ALSO** refers you to other relevant man pages.



More about the man page

- # The NAME section is a *single line* summary of the command, file or function that the page describes, as in:
`man - format and display the on-line manual pages`
- # The SYNOPSIS contains the precise rules for how the command can be used and can be much longer.
- # The DESCRIPTION is a prose description of the use of the command, file, or function.



Searching for commands

- # What if you don't know the name of the command you want?

`man -k keyword`

will search through and list the all man page NAME lines to find those containing the *keyword*.

- # Example:

`man -k perl`

will list *all* man pages containing the word perl, which may number in the hundreds.



Man page options

- # The "-k" is called a *flag* or an *option* to the **man** command. Almost all commands have options. The options part lists these. So does the synopsis, which lists them in two ways. Those that do not need an argument are grouped in square brackets **[]** like this:

```
man [-acdfFhkKtwW]
```

Options can be grouped, or listed separately with preceding hyphens as in:

```
man -ack perl  
man -a -c -k perl
```



Man page options (2)

- ✦ If an option has an optional or required argument, it is listed separately, as in

```
man ... [-M pathlist] [-S sectionlist] ...
```

I can invoke the **man** command using as many options as I need:

```
man -M /usr/share/man/pl:: -S 1:4 vim
```

combines two options with arguments. This will display the **man** page for the **vim** command in Polish!



Using **grep** to help searching

- ⌘ Sometimes the output of **man -k** is too large to examine easily. We can reduce its size usefully by **piping** its output through **more** or through **grep**.
- ⌘ **more** is a command that displays a screen of data at a time. We'll explore it later.
- ⌘ **grep** is a filter program; we give it a pattern as an **argument**, and the name of a file, and it displays lines in that file that match the pattern.
- ⌘ An **argument** to a command is a word that alters the command's output.



More about **grep**

For example:

```
grep 'theory' classnotes
```

will list every line in the file named **classnotes** that has the word **theory** in it.

```
grep 'actg' file1
```

will list every line in **file1** containing the pattern **actg**, anywhere in the line.

theory and **'actg'** are arguments to **grep** above.



Using **grep** to help searching

- **grep** can be used to filter the output of **man**, to show only those lines that match a second pattern:

```
$ man -k perl | grep 'tutorial'  
perldebtut [(1)- Perl debugging tutorial  
. . .
```

which will display the names of man pages that contain various tutorials about Perl.



Other ways to get help

- # There are also special files called *Info documents* on certain UNIX systems. The **info** command can be used to read these documents. Learning how to use the **info** command, however, takes much more time than learning how to use man pages. Read the man page for **info** to learn more.



Exploring a command: **who**

- # The **who** command displays information about who is currently logged in. For example:

```
$ who
nguyen pts/1 Jul 13 14:18 (lab1032)
tbw pts/3 Jul 13 13:49 (blackwoods)
sasa pts/4 Jul 14 15:43 (199.43.48.85)
sweiss pts/5 Jul 14 22:27 (eniac)
$
```



More about **who**

- # **who** outputs several items of information about each user. For now, notice that it includes the *username*, the *time at which the user logged in*, and the *name of the host computer from which the user logged in*.
- # **who** has many options that can change what it displays. Research these and experiment with them.



Another form of **who**: **w**

- Yes, **w** is a one-letter command, short for "**who**", but it displays more than **who**:

```
$ w
22:47:20 up 1 day, 9:01, 3 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU        WHAT
nguyen    pts/1    lab1032-1     Thu14       30:16m     0.45s      0.45s      pine
tbw       pts/3    blackwoods    Thu13       29:05m     0.14s      0.10s      pine
sweiss    pts/6    eniac         22:33       0.00s      0.02s      0.00s      w
$
```



Things to try

- # Look up and try out these simple commands, none of which require any more knowledge of UNIX than you already have.

echo, cat, hostname, logname, last

- # Next read about the **type** command and use it to discover the types of the various commands you now know.

