CSC 7700: Scientific Computing

Module A: Basic Skills

Lecture 1: Survival skills for Unix environments

Dr Frank Löffler

Center for Computation and Technology
Louisiana State University, Baton Rouge, LA

August 30 2013

Unix / Linux

Unix History

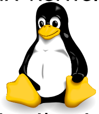
- Originally developed in 1969 by a group of AT&T employees at Bell Labs
- Today's Unix systems split into various branches
- Adoption of Unix by commercial start-ups, most notably Solaris, HP-UX and AIX
- In contrast to: Unix-like operating systems such as Linux and BSD descendants
- Designed to be portable, multi-tasking and multi-user in a time-sharing configuration

Unix Concepts

- Characterized by various concepts:
  - use of plain text for storing data
  - a hierarchical file system
  - treating devices and some forms of inter-process communication as files
  - use of a large number of software tools that can be strung together, as opposed to using a single monolithic program that includes all of the same functionality
- "Operating system" consists of many utilities along with the kernel
- Common executable and Linkable Format (nowadays) : ELF
- Filesystem Hierarchy Standard for common file system layout

GNU/Linux

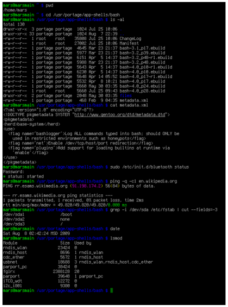
- Unix-like computer operating system using the Linux kernel
- Predominantly known for its use in servers
- Free and open source software collaboration
- Typically packaged in a format known as a Linux distribution, e.g. Fedora, Debian, Ubuntu, openSuse
- Include the Linux kernel and all of the supporting software required to run a complete system
- Main supporting user space system tools and libraries from the GNU Project
- Commonly used software on desktop systems: Firefox, LibreOffice, Gimp, Inkscape



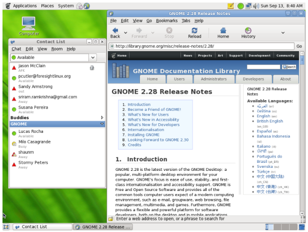
Shells

Shells

- Provides an interface for users of an operating system
- Originates from shells being an outer layer of interface between the user and the internals of the operating system (the kernel)
- Two categories: command-line (CLI) and graphical (GUI)



```
mars@marsmain ~ $ pwd
/home/mars
mars@marsmain ~ $ cd /usr/portage/app-shells/bash
mars@marsmain /usr/portage/app-shells/bash $ ls -al
total 130
drwxr-xr-x  3 portage portage  1024 Jul 25 10:06 .
drwxr-xr-x 33 portage portage  1024 Aug  7 22:39 ..
-rw-r--r--  1 root    root    35808 Jul 25 10:06 ChangeLog
-rw-r--r--  1 root    root    27002 Jul 25 10:06 Manifest
-rw-r--r--  1 portage portage  4645 Mar 23 21:37 bash-3.1_p17.ebuild
-rw-r--r--  1 portage portage  5977 Mar 23 21:37 bash-3.2_p39.ebuild
-rw-r--r--  1 portage portage  6151 Apr  5 14:37 bash-3.2_p48-r1.ebuild
-rw-r--r--  1 portage portage  5988 Mar 23 21:37 bash-3.2_p48.ebuild
-rw-r--r--  1 portage portage  5643 Apr  5 14:37 bash-4.0_p10-r1.ebuild
-rw-r--r--  1 portage portage  6230 Apr  5 14:37 bash-4.0_p10.ebuild
-rw-r--r--  1 portage portage  5648 Apr 14 05:52 bash-4.0_p17-r1.ebuild
-rw-r--r--  1 portage portage  5532 Apr  8 10:21 bash-4.0_p17.ebuild
-rw-r--r--  1 portage portage  5660 May 30 03:35 bash-4.0_p24.ebuild
-rw-r--r--  1 root    root     5660 Jul 25 09:43 bash-4.0_p28.ebuild
drwxr-xr-x  2 portage portage  2048 May 30 03:35 files
-rw-r--r--  1 portage portage   468 Feb  9 04:35 metadata.xml
mars@marsmain /usr/portage/app-shells/bash $ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
<herd>base-system</herd>
<use>
  <flag name='bashlogger'>Log ALL commands typed into bash; should ONLY be
   used in restricted environments such as honeypots</flag>
  <flag name='net'>Enable /dev/tcp/host/port redirection</flag>
  <flag name='plugins'>Add support for loading builtins at runtime via
   'enable'</flag>
</use>
</pkgmetadata>
mars@marsmain /usr/portage/app-shells/bash $ sudo /etc/init.d/bluetooth status
Password:
 * status: started
mars@marsmain /usr/portage/app-shells/bash $ ping -q -c1 en.wikipedia.org
PING rr.esams.wikimedia.org (91.198.174.2) 56(84) bytes of data.

--- rr.esams.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 49.820/49.820/49.820/0.000 ms
mars@marsmain /usr/portage/app-shells/bash $ grep -i /dev/sda /etc/fstab | cut --fields=-3
/dev/sda1               /boot
/dev/sda2               none
/dev/sda3               /
mars@marsmain /usr/portage/app-shells/bash $ date
Sat Aug  8 02:42:24 MSD 2009
mars@marsmain /usr/portage/app-shells/bash $ lsmod
Module                  Size  Used by
rndis_wlan             23424  0
rndis_host              8696  1 rndis_wlan
cdc_ether               5672  1 rndis_host
usbnet                 18688  3 rndis_wlan,rndis_host,cdc_ether
parport_pc             38424  0
fglrx                2388128  20
parport                39648  1 parport_pc
iTCO_wdt               12272  0
i2c_i801                9380  0
mars@marsmain /usr/portage/app-shells/bash $
```



Applications Places System
Sun Sep 13, 8:40 AM
Computer
Contact List
Chat Edit View Room Help
Available
Jason McClain
AFK
pcutler@foresightlinux.org
Available
Sandy Armstrong
out
sriram.ramkrishna@gmail.com
Away
Susana Pereira
Available
Buddies
GNOME
Lucas Rocha
Available
Milo Casagrande
Busy
shaunm
Away
Stormy Peters
Away
GNOME 2.28 Release Notes
File Edit View Go Bookmarks Tabs Help
Back Forward Stop Reload Home History
http://library.gnome.org/misc/release-notes/2.28/
Go
Home News Projects Art Support Development Community
GNOME Documentation Library
Search
Home Users Administrators Developers About
Enter a web address to open, or a phrase to search for
Contact List
GNOME 2.28 Release ...

GNOME 2.28 Release Notes

1. Introduction
2. Become a Friend of GNOME!
3. What's New for Users
4. What's New in Accessibility
5. What's New for Developers
6. Internationalisation
7. Installing GNOME
8. Looking Forward to GNOME 2.30
9. Credits

1. Introduction

GNOME 2.28 is the latest version of the GNOME Desktop: a popular, multi-platform desktop environment for your computer. GNOME's focus is ease of use, stability, and first-class internationalisation and accessibility support. GNOME is Free and Open Source Software and provides all of the common tools computer users expect of a modern computing environment, such as e-mail, groupware, web browsing, file management, multimedia, and games. Furthermore, GNOME provides a flexible and powerful platform for software developers, both on the desktop and in mobile applications

GNOME 2.28 Release Notes

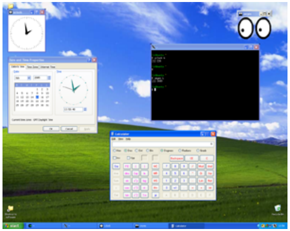
Available Languages:
- العربيّة (ar)
- čeština (cs)
- English (en)
- British English (en_GB)
- Español (es)
- Bahasa Indonesia (id)
- Italiano (it)
- ਪੰਜਾਬੀ (pa)
- Português do Brasil (pt_BR)
- Svenska (sv)
- Türkçe (tr)
- 中文 (中国大陆) (zh_CN)
- 中文 (香港) (zh_HK)
- 中文 (台灣) (zh_TW)

CLI Shells

- Often simply called “shells”
- Provide a command-line interface (CLI)
- Mechanism for interacting with a computer operating system or software by typing commands to perform specific tasks
- Text-only interface
- Command-line interpreter receives, analyzes, and executes requested commands, examples: sh, ksh, csh, tcsh, bash
- Can be embedded in GUIs
- Most popular: BASH

GUIs in Unix/Linux

- Using X Window System
  - Provides graphics usage and manages keyboard and mouse control functions
  - Does not mandate the user interface (task of Window managers)
  - Specifically designed to be used over network connections
  - Can be tunneled, e.g. via Secure Shell
  - Lots of implementations, e.g. XFree86, X.Org, DECwindows, MacX, Cygwin/X, Exceed
- Window managers handle design of interface, e.g. Metacity, KWin, Enlightenment, Compiz, Xfce, Xfwm



CLI Essentials

Basic commands

Function	Command
Show current directory name	`pwd`
List directory content	`ls [files / directories]`
Change current directory	`cd [directory name]`
Create directory	`mkdir <directory name(s)>`
Remove directory	`rmdir <directory name(s)>`
Remove files	`rm <file name(s)>`
Rename/move file or directory	`mv <source> <destination>`
Logout / exit	`logout`/`exit` (typically `Crtl+d`)

CLI Essentials

Filesystem path essentials

Directory separator	`/`
Current directory	`.`
Parent directory	`..`
Home directory (*)	`~`
Previous directory (*)	`-` (to be used as `cd -`)
Wildcard 'any character'(*)	`?`
Wildcard 'any number of characters' (*)	`*`
Configuration files	`~/.?*`
Allowed characters in filename	anything but `\0` and `/`
Case sensitive!	`README` ≠ `readme`
No (real) filename extensions	

(*) Shell-specific

CLI Essentials

Special shell characters

Character	Meaning
#	Comment until end of line
;	Command separator
"	Partial quoting
'	Full quoting
\	Escaping
*	Wildcard 'any number of characters'
?	Wildcard 'any character'
$	Variable substitution

CLI Essentials

Stdin, stdout and pipes

Stdin, stdout: "standard in" and "standard out": input and output streams, usually connected to keyboard and display.

Character	Meaning
`cmd > file`	Send stdout of `cmd` to file
`cmd < file`	Send contents of file as stdin of `cmd`
`cmd1 | cmd2`	Pipe stdout of `cmd1` to stdin of `cmd2`
`-`	Often used shorthand for stdin or stdout
`<file1 cmd1 | cmd2 > file2`	Example of combination

Text Editors

Text Editors

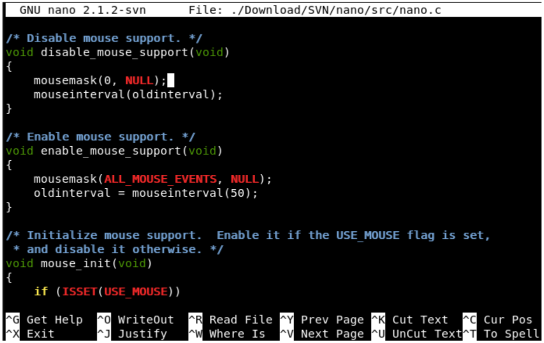
- Program used for editing plain text files (as opposed to e.g. Word documents)
- Examples: Vi, Emacs, nano, Notepad, TextEdit
- Typical features
  - String searching algorithm
  - Cut, copy, and paste
  - Text formatting
  - Undo and redo
  - Syntax highlighting
- Specialized Editors for e.g. Source code, IDEs, HTML, TeX

Plain Text Files

- Structured as a sequence of lines
- End of a text file is sometimes denoted by a special characters (EOF), system-dependent
- End of line is indicated by EOL marker
    - `LF`: Unix, Linux, Max OS X, BeOS, Amiga
    - `CR+LF`: DOS, Windows, OS/2, Symbian
    - `CR`: MacOS up to version 9
- Character encodings, e.g.:
    - ASCII: simple, but very limited
    - ISO 8859-?: Extensions of ASCII, but still very limited
    - UTF-8: backwards compatible to ASCII, but very large character set
- EOL and encoding conversions sometimes necessary

Text Editor Examples: Nano

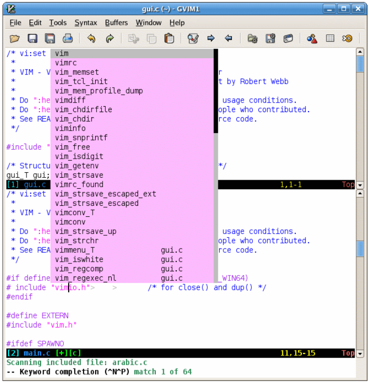
- Using text terminal
- Free replacement for earlier editor 'pico' (included in 'pine')
- Very easy, but also very basic



```
  GNU nano 2.1.2-svn       File: ./Download/SVN/nano/src/nano.c

/* Disable mouse support. */
void disable_mouse_support(void)
{
    mousemask(0, NULL);
    mouseinterval(oldinterval);
}

/* Enable mouse support. */
void enable_mouse_support(void)
{
    mousemask(ALL_MOUSE_EVENTS, NULL);
    oldinterval = mouseinterval(50);
}

/* Initialize mouse support.  Enable it if the USE_MOUSE flag is set,
 * and disable it otherwise. */
void mouse_init(void)
{
    if (ISSET(USE_MOUSE))

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Text Editor Examples: Vim

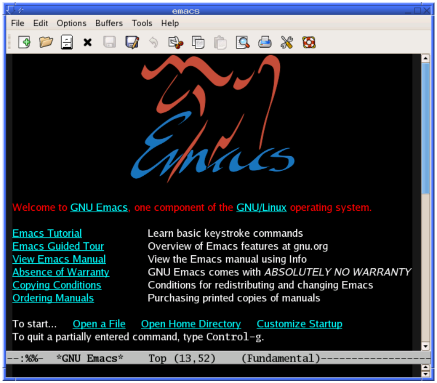
- Modal editing (insert-mode and command mode) by switching entire keyboard in and out of modes
- Can (but does not have to be) used entirely with the keyboard
- Minimal use of Meta keys
- Can be extensively customized
- Available for virtually every operating system



```
/* vi:set  vim
 *        vimrc
 * VIM - V vim_memset                                   r
 *        vim_tcl_init                                 t by Robert Webb
 *        vim_mem_profile_dump
 * Do ":he vimdiff                                      usage conditions.
 * Do ":he vim_chdirfile                               ople who contributed.
 * See REA vim_chdir                                   rce code.
 */       viminfo
          vim_snprintf
#include " vim_free
          vim_isdigit
/* Structu vim_getenv                                  */
gui_T gui; vim_strsave
[1] gui.c  vimrc_found                                          1,1-1        Top
/* vi:set  vim_strsave_escaped_ext
 *        vim_strsave_escaped
 * VIM - V vimconv_T
 *        vimconv
 * Do ":he vim_strsave_up                               usage conditions.
 * Do ":he vim_strchr                                  ople who contributed.
 * See REA vimmenu_T                  gui.c            rce code.
 */       vim_iswhite                 gui.c
          vim_regcomp                 gui.c
#if define vim_regexec_nl              gui.c            _WIN64)
# include "vimio.h">    >        /* for close() and dup() */
#endif

#define EXTERN
#include "vim.h"

#ifdef SPAWNO
[2] main.c [+][c]                                                11,15-15     Top
Scanning included file: arabic.c
-- Keyword completion (^N^P) match 1 of 64
```

Text Editor Examples: Emacs

- Very feature-rich
- Highly customizable
- Extensive use of Meta keys
- Available for virtually every operating system



Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial	Learn basic keystroke commands
Emacs Guided Tour	Overview of Emacs features at gnu.org
View Emacs Manual	View the Emacs manual using Info
Absence of Warranty	GNU Emacs comes with *ABSOLUTELY NO WARRANTY*
Copying Conditions	Conditions for redistributing and changing Emacs
Ordering Manuals	Purchasing printed copies of manuals

To start... Open a File Open Home Directory Customize Startup
To quit a partially entered command, type `Control-g`.

```
--:%%-  *GNU Emacs*    Top (13,52)    (Fundamental)----------------
```

Text Editor Recommendations

- No single recommendation (There isn't **the** text editor.)
- Choose whatever **you** like
- Important aspects: Use editor that can
  - handle UTF-8 encoding (and use it)
  - understand and respect different EOL styles
- Nice-to-have aspects
  - Syntax-highlighting
  - Available on multiple OSs
- All three recommendations (nano, vim, emacs) fulfill above points, but many others do as well
- Again: choose whatever **you** like

Secure SHell

Uses of SSH

- Login to a shell on a remote host
- Executing a single command on a remote host
- Secure file transfer
- In combination with rsync to back up, copy and mirror files efficiently and securely
- Forwarding or tunneling a network port
- Using as a full-fledged, encrypted VPN.
- Securely forwarding X from a remote host
- Browsing the web through an encrypted proxy connection (SOCKS)
- Securely mounting a directory on a remote server as a filesystem on a local computer (SSHFS)

Early history

- 1995: Tatu Ylönen, a researcher at Helsinki University of Technology, Finland, designed the first version of the SSH protocol
- Reason: password-sniffing attack at his university network
- Goal: replace the earlier rlogin, TELNET and rsh protocols
- End of 1995: about 20.000 users in 50 countries (freeware)
- Releases evolved into increasingly proprietary software
- 1999: Björn Grönvall forked old, free version of SSH
- Shortly thereafter, OpenBSD forked Grönvall's code, creating OpenSSH
- Since 2005: OpenSSH is the single most popular SSH implementation

Recent History

- 2006: Version 2 of SSH protocol: SSH-2
- Incompatible with version 1
- Both security and feature improvements
- OpenSSH implements both versions
- By default, version 1 is usually disabled

Internet standards

- RFC 4250, The Secure Shell (SSH) Protocol Assigned Numbers
- RFC 4251, The Secure Shell (SSH) Protocol Architecture
- RFC 4252, The Secure Shell (SSH) Authentication Protocol
- RFC 4253, The Secure Shell (SSH) Transport Layer Protocol
- RFC 4254, The Secure Shell (SSH) Connection Protocol
- RFC 4255, Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints
- RFC 4256, Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)
- RFC 4335, The Secure Shell (SSH) Session Channel Break Extension
- RFC 4344, The Secure Shell (SSH) Transport Layer Encryption Modes
- RFC 4345, Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol
- RFC 4419, Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4432, RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4462, Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol (May 2006)
- RFC 4716, The Secure Shell (SSH) Public Key File Format (November 2006)
- RFC 5656, Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer (December 2009)

Authentication

Authentication

- Establishing or confirming something (or someone) as authentic
- Not to be confused with Authorization
  - Authentication must precede authorization
- Short notations: A1/AuthN (authentication), A2/AuthZ (authorization)
- Impossible to prove the identity of a computer user with absolute certainty

Mechanism examples:

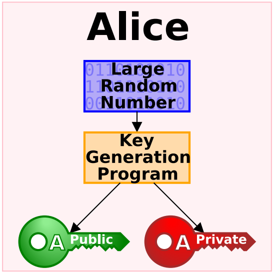
- Shared secret (e.g. passwords, key cards)
- Public+secret key pairs
- Certificates

Public-Key Cryptography

Public-Key Cryptography

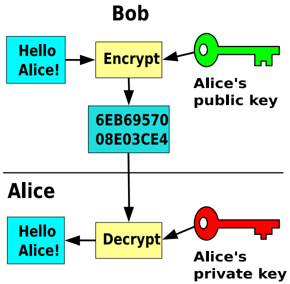
- Use of asymmetric key algorithms
- No need for secure preceding secret key exchange (unlike for symmetric key algorithms)
- Creation of mathematically related key pair:
  - secret, private key
  - published, public key
- Can be used for protection of
  - confidentiality and integrity by public key encryption
  - authenticity by digital signatures

Asymmetric key algorithm



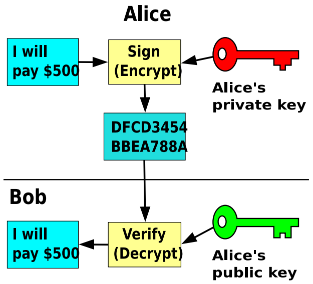
An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.

Asymmetric key algorithm



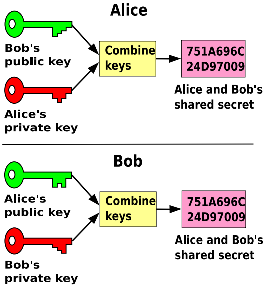
Anyone can encrypt messages using the public key, but only the holder of the paired private key can decrypt. Security depends on the secrecy of that private key.

Asymmetric key algorithm



The private key can be used to sign a message; but anyone can check the signature using the public key. Validity depends on private key security.

Asymmetric key algorithm



In the Diffie–Hellman key exchange scheme (which is used in SSH), each party generates a public/private key pair and distributes the public key. After obtaining an authentic copy of each other's public keys, Alice and Bob can compute a shared secret. The shared secret can be used as the key for a symmetric cipher.

Asymmetric key algorithm

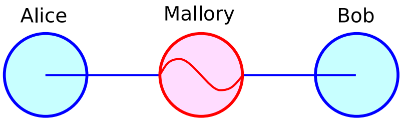
Central problem:

- Confidence that a public key is correct, belongs to the person or entity claimed, and has not been tampered with or replaced by a malicious third party

Usual approaches:

- "Web of trust" like used in e.g. PGP (not used for SSH)
- Certificates and certificate authorities

Man in the middle attack



1. Alice sends a message to Bob, which is intercepted by Mallory:
   Alice *"Hi Bob, it's Alice. Give me your key"* → **Mallory Bob**
2. M. relays message to Bob, who cannot tell its not from Alice:
   Alice Mallory "Hi Bob, it's Alice. Give me your key." → **Bob**
3. Bob responds with his encryption key:
   Alice Mallory ← [Bob's key] **Bob**
4. Mallory replaces Bob's key with own, and relays:
   Alice ← [Mallory's key] **Mallory Bob**
5. Alice encrypts a message with that key:
   Alice "Buy stock A." [Mallory's key] → **Mallory Bob**
6. Mallory decrypts, reads, modifies, re-encrypts and forwards:
   Alice Mallory "Sell everything." [Bob's key] → **Bob**

File transfer protocols within SSH

- Secure copy (SCP): rcp replacement
  - Combination of RCP and SSH
  - Only transfer of files
  - Usually command line clients
- SSH File Transfer Protocol (SFTP)
  - secure alternative to FTP
  - resuming, directory listings, remote file removal
  - Gui clients feasible

Above can be used by either dedicated clients, or by e.g. SSHFS.

- rsync
  - Not implemented within SSH itself, nor using above methods
  - Uses SSH tunnel to synchronize data between machines

SSH implementation examples

SSH Server implementation examples

- Some commercial implementations
- OpenSSH
  - By far the most popular implementation
  - Available on almost every platform, from PC over Mac, Amiga, to mobile phones
  - Feature-rich
- CopSSH
  - SSH server package for Microsoft Windows
  - uses OpenSSH, OpenSSL, Cygwin and GNU tools
- Dropbear
  - lightweight SSH-2 implementation
  - designed for environments with low memory and processor resources
  - no SSH-1 or SFTP support

SSH client implementation examples

- Some commercial implementations
- OpenSSH
  - Standard tool in most Linux/Unix distributions
- PuTTY
  - Originally a Microsoft Windows GUI client
  - Now ported to a number of other OSs as well
- Dropbear
- CopSSH
- ...

Basic login commands

Simplest command:

```
ssh hostname
```

Forwarding X:

```
ssh -X user@hostname
```

Executing single command:

```
ssh user@hostname.domain ls
```

Using a “trampoline” (there are better ways):

```
ssh -t user1@hostname1 ssh user2@hostname2
```

Using above to open remote X terminal:

```
ssh -X user@hostname1 xterm
```

Key management

(Some) Authentication options

- Passwords
- Key pairs
- Certificates

Key pairs:

- Public/Secret, personal key pair
- Used instead of password
- Secret key usually passphrase-protected
- Two (three) key types: rsa, dsa (rsa1 for ssh-1)
- Usually stored in `$HOME/.ssh/id_dsa[.pub]` / `$HOME/.ssh/id_rsa[.pub]`

Key management examples

Tool to create and change keys: `ssh-keygen`, usage examples:
Creating new key:

```
ssh-keygen
```

Changing the passphrase of an existing key:

```
ssh-keygen -p
```

Authorizing key as replacement for password by adding public key to remote file

```
cat ~/.ssh/id_rsa.pub | ssh user@machine.domain \
  'mkdir -p .ssh; cat >> .ssh/authorized_keys'
```

SSH Agents

Private SSH keys should have passphrase set (for most uses). But:

- Cumbersome to always enter passphrases
- Even more problematic when "hopping" from host to host

Solution: SSH-Agents

- Will ask for password once, (securely) store it and re-use later
- Configurable lifetime of stored passwords
- Tools: `ssh-agent`, `keychain`

SSH can forward agent information:

```
ssh -A user@hostname
```

Port Forwarding

Tunnel network connection through SSH connection
Examples:

- Reach hostB:22 “directly” as localhost:2222 through hostA
```
ssh -L 2222:hostB:22 hostA
```
- Reach one host (localhost:80) from remote host (port 8080)
```
ssh -R 8080:localhost:80 host
```
- Browse using secure tunnel to host (SOCKS)
```
ssh -D localhost:8080 host
```

Config File

```
$ cat ~/.ssh/config
ServerAliveInterval 10
ServerAliveCountMax 60

  Host hostname.somewhere alias
    IdentityFile ~/.ssh/somekey/id_rsa
    User         myothername
    ForwardX11   yes
    ProxyCommand ssh user@gateway.hostname.tld nc %h %p
```

Summary

Summary

Secure Shell (SSH)

- Provides secure network channel for variety of uses, e.g:
    - Remote login shell
    - Secure file transfer
    - Network traffic tunneling

Best usage practices:

- Use ssh keys instead of passwords
- Set passphrase for ssh keys
- Use ssh agent to ease work with ssh keys
- Use ProxyCommand for ssh-“hopping”