CS615 - Aspects of System Administration

Software Installation Concepts / Multiuser Fundamentals

Department of Computer Science
Stevens Institute of Technology
Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/615/
Hooray!

5 Minute Break
Software Installation Concepts

Operating System Installation
OS Installation

NetBSD/amd64 6.1.3

This menu-driven tool is designed to help you install NetBSD to a hard disk, or upgrade an existing NetBSD system, with a minimum of work. In the following menus, type the reference letter (a, b, c, ...) to select an item, or type CTRL+M/CTRL+P to select the next/previous item. The arrow keys and Page-up/Page-down may also work. Activate the current selection from the menu by typing the Enter key.

Thank you for using NetBSD!

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Install NetBSD to hard disk</td>
</tr>
<tr>
<td>b</td>
<td>Upgrade NetBSD on a hard disk</td>
</tr>
<tr>
<td>c</td>
<td>Re-install sets or install additional sets</td>
</tr>
<tr>
<td>d</td>
<td>Reboot the computer</td>
</tr>
<tr>
<td>e</td>
<td>Utility menu</td>
</tr>
<tr>
<td>f</td>
<td>Config menu</td>
</tr>
<tr>
<td>x</td>
<td>Exit Install System</td>
</tr>
</tbody>
</table>
OS Installation

```
# fdisk -f -u 0 -s 169/63/4194241 /dev/rwd0d
# fdisk -f -c /usr/mdec/mbr /dev/rwd0d
# fdisk -f -a -0 /dev/rwd0d
# disklabel -e -I wd0
[...]
4 partitions:
#             size  offset fstype [fsize bsise cpg/sgs]
 a:  4194241  63  4.2BSD  0  0  0 # (Cyl. 0*-- 4161*)
 c:  4194241  63  4.2BSD  0  0  0 # (Cyl. 0*-- 4161*)
 d:  4194304  0  unused  0  0  0 # (Cyl. 0 - 4161*)
# /sbin/newfs -O 2 /dev/rwd0a
/dev/rwd0a: 2048.0MB (4194240 sectors) block size 16384,
    fragment size 2048 using 12 cylinder groups of
    170.67MB, 10923 blks, 21504 inodes.
super-block backups (for fsck_ffs -b #) at:
    32, 349568, 699104, 1048640, 1398176, 1747712, 2097248, 2446784,
..............................................................
# mount -o async /dev/wd0a /mnt
# for pkg in base comp etc games man misc modules text kern~GENERIC; do
tar zxf /i386/binary/sets/\{pkg\}.tgz -C /mnt
done
# cp /mnt/usr/mdec/boot /mnt/boot
# /usr/sbin/installboot -v -o timeout=5 /dev/rwd0a
    /mnt/usr/mdec/bootxx_ffsv2
File system: /dev/rwd0a
Primary bootstrap: /usr/mdec/bootxx_ffsv2
Boot options:    timeout 5, flags 0, speed 9600, ioaddr 0, console pc
# cd /mnt/dev /MKDEV all
# shutdown -r now
```
OS Installation

General steps:

- power up
- PXE or iPXE boot
  - network configuration / BOOTP/DHCP
  - boot from network via e.g., tftp miniroot
- identify root device and optional additional disks
- create partition table / disklabel
- create filesystem(s)
- install MBR, bootblocks etc.
- fetch OS software (e.g., via HTTPS, iSCSI, ...)
- install / copy / extract OS
- optionally add application software
- perform basic system configuration
- reboot
OS Installation

Most of the difficult parts happen outside of the building system:

- hardware identification, provisioning, and registration
- base OS installation
- installation of add-on applications
- initial minimum system configuration [*]
- system registration
- system restart

[*] system *deployment* ∩ system *configuration*  
⇒ configuration management
Post Installation
Software Installation Concepts

System Software vs. Third Party Software
## System Software vs. Third Party Software

<table>
<thead>
<tr>
<th>Example</th>
<th>System / OS</th>
<th>3rd Party</th>
<th>Packaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drivers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>firmware</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shell</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>compiler</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssh(1) / sshd(8)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mail server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>web server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>python</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Types of Software

- **Add-on or Third-Party Applications**
  - (web browser, database, programming languages, ...)

- **System Software**
  - (device drivers, loadable modules, libraries, ...)

- **Applications/Utilities**
  - (shell, common unix tools, daemons, compiler, ...)

- **Kernel**

- **Firmware**

- **Hardware**
System Software vs. Third Party Software

Consider:

- OS upgrades vs. software upgrades
- location of configuration files
- duplicates or conflicting versions in the base system vs. the add-ons
- startup scripts, dæmons
- location of third party software
- dependencies
- installation by hand and/or installation using a package manager
- proprietary third party software
Package Manager Features

- easy and scalable installation of software
- automatic resolution of software dependencies
- package and file inventory

```bash
linux-lab$ dpkg -l
[...]
linux-lab$ dpkg -L tcpdump
[...]
linux-lab$ dpkg-query -S /usr/lib/libDeployPkg.so.0
[...]
```
Package Manager Features

- easy and scalable installation of software
- automatic resolution of software dependencies
- package and file inventory
- integration into OS
- package and file integrity checks

```
$ rpm -Va
[...]
missing /etc/pki/CA/private (Permission denied)
S.5..... c /etc/pki/tls/certs/ca-bundle.crt
.......T c /etc/libuser.conf
..?..... c /etc/tcsd.conf
missing c /etc/logrotate.d/syslog
[...]
```
Managing Security Patches and Software Upgrades

How many known vulnerabilities (unique CVEs and affected packages) exist in each of the Fedora and Debian instances?

debian$ sudo apt-get install debsecan
debian$ debsecan
debian$ sudo apt-get update
debian$ sudo apt-get upgrade
debian$ debsecan

defedora$ yum list-security
ndefedora$ yum info-security
Team Missions
Managing Security Patches and Software Upgrades

How many known vulnerabilities (unique CVEs and affected packages) exist in each of the Fedora and Debian instances?

debian$ sudo apt-get install debsecan
debian$ debsecan
debian$ sudo apt-get update
debian$ sudo apt-get upgrade
debian$ debsecan

defora$ yum list-security
defora$ yum info-security
defora$ sudo yum update
defora$ yum list-security

Excellent! Now what about all the stuff you installed that wasn’t packaged?
“What’s pip?”
“A python package manager”
“How do I install it?”
“easy_install pip”
“What’s easy_install?”
“A python package manager”
Special Purpose Package Managers

"What is Bower?"
"A package manager"
"How do I install it?"
"Use npm"
"What's npm?"
"A package manager"
"...."
Special Purpose Package Managers

Most programming languages or environments come with their own "package management" solutions, often integrating/mixing with a "build system".

- Common Lisp => quicklisp
- Go => go get
- NodeJS => npm
- Perl => CPAN
- Python => easy-install, pip, pants, setuptools, ...
- Ruby => gems, rvm, rake
- Scala => sbt
- ...

...
You don’t get to choose.

You routinely have to build from source and (re-)package your software.
Dependencies, Integrity, and Trust

OS provider repositories:
- `yum update/yum install`
- `apt-get`

Language-specific community repositories:
- `gem install foo`
- `go get github.com/randomAccount/randomRepository`
- `npm install -g foo`
- `perl -MCPAN -e 'install Something::YouWant'`
- `pip install foo`

What could possibly go wrong?
Dependencies, Integrity, and Trust

Fun fact:

$ wget http://somewhere/some.tar.gz
$ tar zxf some.tar.gz
$ cd some
$ ./configure
$ make
$ sudo make install

is not inherently better than

$ curl http://somewhere/script.sh | sudo bash
Dependencies, Integrity, and Trust

Mirroring untrusted, unverified dependencies does not solve any of your problems.

Integrity verification is meaningless without assurance of trust.

Dependencies are called dependencies because you depend on them.

Dependency trust and integrity is recursive.
Dependencies, Integrity, and Trust

Mirroring untrusted, unverified dependencies does not solve any of your problems.

Integrity verification is meaningless without assurance of trust.

Dependencies are called dependencies because you depend on them.

Dependency trust and integrity is recursive.

*Remember Left-Pad!*
Exercises

All recommended, none graded:
https://stevens.netmeister.org/615/package-exercise.html

Identify a piece of software you use, but that's not packaged for a given package manager. Create a package for it, then contribute upstream.

Create a cheat sheet for 4 different package managers, listing the more important equivalent commands.

How does your preferred OS update firmware?

How does the concept of *reproducible builds* relate to what we discussed here?

What is the overlap with system *configuration*? Can a package manager assert state?
Hooray!

5 Minute Break
Multiuser

UNIX was designed from the beginning (1970s) as a portable, multi-tasking, *multi-user* system.

Windows gained this functionality with WindowsNT in 1993.

Mac OS followed in 2001 with OS X.
Implications of a Multi-User System
Implications of a Multi-User System
Consider Scalability

Things to consider:
Granting Privileges requires Trust

- different environments have different trust models
- human interactions in small groups strengthen trust
- larger groups are divided into smaller, close-knit groups
- the more groups you have, the weaker their trust bonds are
Granting Privileges requires Trust

- different environments have different trust models
- human interactions in small groups strengthen trust
- larger groups are divided into smaller, close-knit groups
- the more groups you have, the weaker their trust bonds are

Trust does not scale.
Granting Privileges requires Trust

We are considering computer-human systems.

For humans, trust, but (be able to) verify.

For computers, apply the Least Privilege principle.
Implications of a Multi-User System

- users may want to keep files private
Implications of a Multi-User System

- users may want to keep files private
- users may want to share files
Implications of a Multi-User System

- users may want to keep files private
- users may want to share files
- users may (try to gain) access to files they shouldn’t have access to
Implications of a Multi-User System

- users may want to keep files private
- users may want to share files
- users may (try to gain) access to files they shouldn’t have access to
- users may (want to) do things that affect other users
Implications of a Multi-User System

- users may want to keep files private
- users may want to share files
- users may (try to gain) access to files they shouldn’t have access to
- users may (want to) do things that affect other users
- different users may require different privileges
Users and User-IDs

Bijective?

alice
bob
claire
dennis
edsger
root
nobody
Users and User-IDs

Not surjective!
Users and User-IDs

Not injective, either!
Users and User-IDs

nobody
Authentication

- proof of identity, not proof of *authorization*
Authentication

- proof of identity, not proof of *authorization*
- something you know, something you have, something you are
Authentication

- proof of identity, not proof of *authorization*
- something you know, something you have, something you are
- multi-factor authentication combines these to help protect against different threats
Authentication

- proof of identity, not proof of *authorization*
- something you know, something you have, something you are
- multi-factor authentication combines these to help protect against different threats
- mutual authentication may be a requirement
Authentication

Common examples:

NetBSD/amd64 (SERVER) (console)

login: jschauma
password: *********************************
NetBSD 7.0.2 (SERVER) #2: Tue Jan 24 02:33:13 EST 2017

Welcome to NetBSD!
hostname$
Authentication

Common examples:

$ ssh-keygen -l -f /dev/stdin <<<$(aws ec2 get-console-output \
                       i-0990f1eb069c853c4 | grep ^ecdsa)
$ ssh -i ~/.ssh/myawskey ec2-54-227-16-184.compute-1.amazonaws.com
The authenticity of host 'ec2-54-227-16-184.compute-1.amazonaws.com
(54.227.16.184)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
NetBSD 7.0.2 (SERVER) #2: Tue Jan 24 02:33:13 EST 2017

Welcome to NetBSD!
hostname$
Authentication

Common examples:

$ kinit
Password for jschauma@DOMAIN: ******************************

$ klist
Ticket cache: /tmp/krb5cc_ttypa
  Default principal: jschauma@DOMAIN

  Valid starting   Expires             Service principal
  02/13/17 13:50:21 02/13/17 21:50:20  krbtgt/KDC@DOMAIN

$ ssh somehost
somehost$
Authentication

Common examples:

```
localhost$ ssh sshca
YubiKey for ‘jschauma’: **********************
Password: **********************
localhost$ ssh-add -l
2048 SHA256:TzwuHGc5BKBe+VJSnGoVyh92J8XKBuKaL7MGQn8ML0Y (RSA)
2048 SHA256:TzwuHGc5BKBe+VJSnGoVyh92J8XKBuKaL7MGQn8ML0Y (RSA-CERT)
localhost$ ssh somehost
Duo two-factor login for jschauma

Enter a passcode or select one of the following options:

1. Duo Push to XXX-XXX-0712
2. Phone call to XXX-XXX-0712
3. SMS passcodes to XXX-XXX-0712

Passcode or option (1-3): 1
Success. Logging you in...
Last login: Thu Jan 26 17:39:30 2017 from 10.1.2.3
```

somehost$
Authentication

Common examples:
Authentication

Common examples:
- passwords, PINs
- ssh keys, PGP keys, X.509 certificates
- security tokens: OTPs in hardware or software, RFIDs
- physical biometrics: fingerprint, retina scan, facial recognition
- behavioral biometrics: speech pattern, gait, keystroke dynamics...

Mix and match the above to yield multi-factor authentication:
- password + PIN via e.g. SMS
- ssh key + TOTP from e.g. mobile device
- fingerprint + security token
- ...
UNIX Fundamentals: User Accounts and File Permissions

Every account
- has a *unique* ID
- belongs to at least one group
- may or may not be password protected
- may or may not have a valid login program
- may or may not be allowed to escalate privileges
UNIX Fundamentals: User Accounts and File Permissions

Every account
- has a *unique* ID
- belongs to at least one group
- may or may not be password protected
- may or may not have a valid login program
- may or may not be allowed to escalate privileges

Every file
- is associated with a *uid* and a *gid*
- has a number of protection bits
UNIX Fundamentals: User Accounts and File Permissions

The image shows a file listing with permissions and details:

```
-rw-r--r-- 1 root wheel 1396 Aug 17 08:59 /etc/passwd
```

- **file name**: `/etc/passwd`
- **last modified date**: Aug 17 08:59
- **size in bytes**: 1396
- **group**: root
- **owner**: wheel
- **number of hard links**: 1
- **execute permissions for 'other'**: -
- **write permissions for 'other'**: -
- **read permissions for 'other'**: -
- **execute permissions for 'group'**: -
- **write permissions for 'group'**: -
- **read permissions for 'group'**: -
- **execute permissions for 'owner'**: -
- **write permissions for 'owner'**: -
- **read permissions for 'owner'**: -
- **file type**: -
Raising privileges

Some tasks require special privileges:

- binding a port < 1024 (e.g. 22, 25, 80, 443)
- operating on raw sockets (e.g. ping(1), traceroute(8))
- changing local passwords
- accessing files/directories without explicit permissions
- just about anything involving file systems
- ...
Raising privileges

Options:

somehost$ exit

$ ssh root@somehost

#
Raising privileges

Options:

$ su user2 -c 'some command'
Password:
$ su - root
Password:
#
Raising privileges

Options:

somehost$ sudo bash
jschauma is not allowed to run sudo on somehost. This incident will be reported.
Raising privileges

Options:

```
jshauma@somehost$ ls dir
ls: cannot open directory dir: Permission denied
jshauma@somehost$ sudo bash
Sorry, user jshauma is not allowed to execute '/bin/bash' as root on somehost.
jshauma@somehost$ sudo ls dir
Sorry, user jshauma is not allowed to execute '/bin/ls' as root on somehost.
jshauma@somehost$ sudo -u otheruser ls dir
Password: ****************************
file1  file2
jshauma@somehost$
```
Unix Groups

- enables *arbitrary* collections of users to share resources
- information stored in `/etc/group`, format is:
  - `name:*:GID:user1,user2,...`
- most Unix systems impose a limit of 16 or 32 group memberships per user
- most Unix systems have a common default group for new users (some Linux versions deviate)
- some Unix systems have/had group shadow files
Group Access

At any but the smallest environments, we find:
- a central user database
- users divided into different access groups
- access to systems is granted primarily by such group membership
- privileges on a system are also granted by such group membership

The privileges granted in this manner are commonly broken down and controlled via *role-based access control* (RBAC).
Group Access
Multiuser Truths

- All users are equal.
- Some users are more equal than others.
- The principle of least privilege applies to all.
- Humans require trust.
- Trust does not scale.
- You will always face trade-offs.
Adding and Removing Accounts

Recommended exercise:
https://stevens.netmeister.org/615/useradd-exercise.html
Reading

User Management:

- Frisch: Ch 6; Burgess: Ch 5;

- https://is.gd/wg50sE