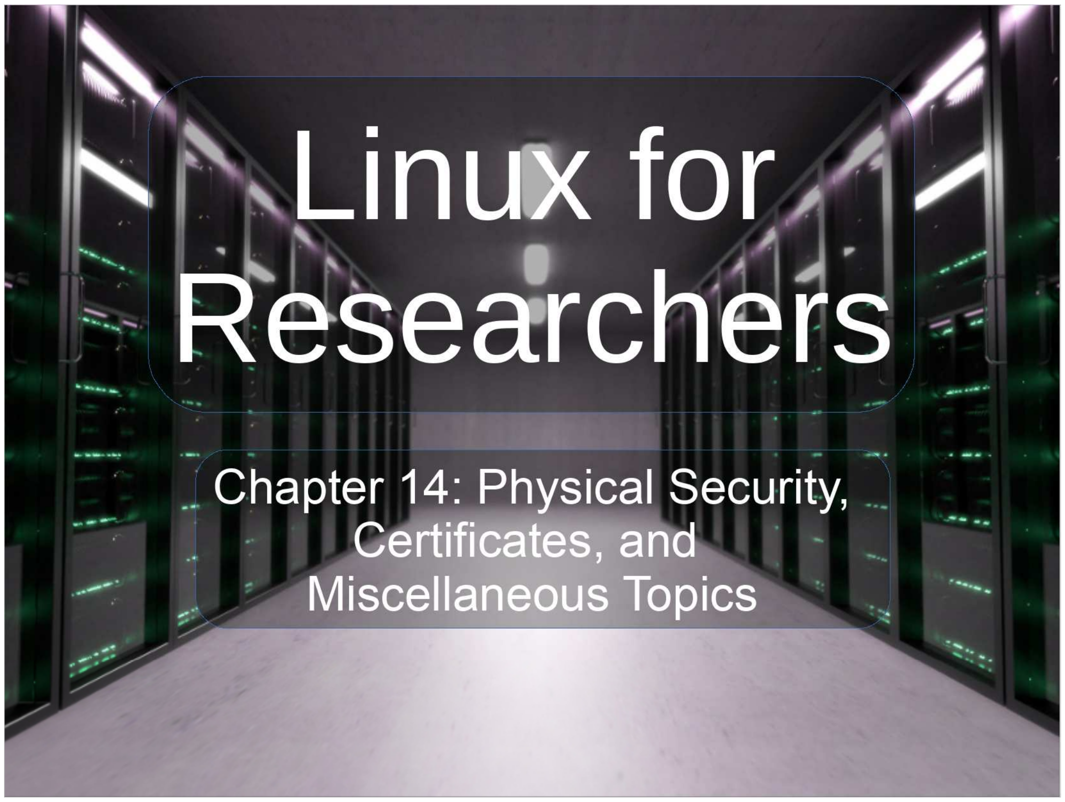


Linux for Researchers

Chapter 14: Physical Security, Certificates, and Miscellaneous Topics

This is the last talk in this series. Today we'll talk about two biggish topics: the importance of physical security, and a quick overview of certificates. Then we'll run though several short topics that we just haven't been able to fit in earlier.

Part 1: Physical Security

Now that you're an experienced Linux system administrator, what would you do if someone brought a computer to you and told you that they'd lost their root password and couldn't log in?

Let's look at a few ways you could break in.

Using the Grub Menu to Break in:

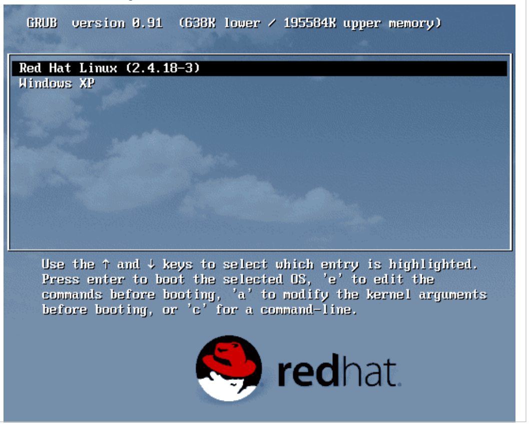
Unless the Grub menu is password-protected, you can press “e” while the menu is displayed and alter the highlighted menu entry's configuration. For example, this entry:

```
title CentOS (2.6.18-92.1.22.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-92.1.22.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-92.1.22.el5.img
```

could be changed by adding “single” to the end of the “kernel” line. This would cause the computer to boot into single-user mode. In this mode, you'll usually be logged in as root automatically, without the need to for a password.

If that doesn't work, you could add “single init=/bin/sh” instead.

This would bypass any security barriers that the operating system might present, and leave you with a command prompt, and operating as the root user.



```
GRUB  version 0.91  (638K lower / 195584K upper memory)

Red Hat Linux (2.4.18-3)
Windows XP

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, 'a' to modify the kernel arguments
before booting, or 'c' for a command-line.
```

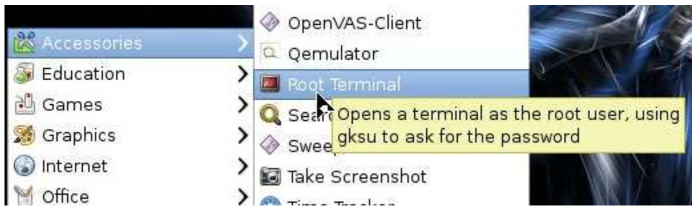
Using “init=/bin/sh” just tells the kernel to start /bin/sh as the first process, instead of the usual /sbin/init. You could have it run any program you want, but /bin/sh will give you a shell prompt, from which you can do whatever you need to do. Embedded Linux systems (in phones, routers, toasters, etc.) often have the kernel invoke a specialized, simple, init program instead of the general-purpose /sbin/init.

Note that, if you use “init=/bin/sh”, you may find that the root filesystem is mounted read-only. You'll notice this when you try to modify a file, and find that you can't. This can be easily fixed by remounting the filesystem to make it writeable:

```
mount -o remount,rw /
```

Breaking in Using Removable Media:

If you can't break in through Grub, try a KNOPPIX DVD. After you've booted KNOPPIX, click the application menu a the bottom and open a root shell. From here, you'll be able to mount the computer's drives and do anything you need to do.



Opens a terminal as the root user, using gksu to ask for the password

Note that, if the computer uses Logical Volumes (LVM), you may need to enable them by hand:

```
[root@demo ~]# vgscan
[root@demo ~]# vgchange -a y
```

Scan for logical volumes

Activate all logical volumes

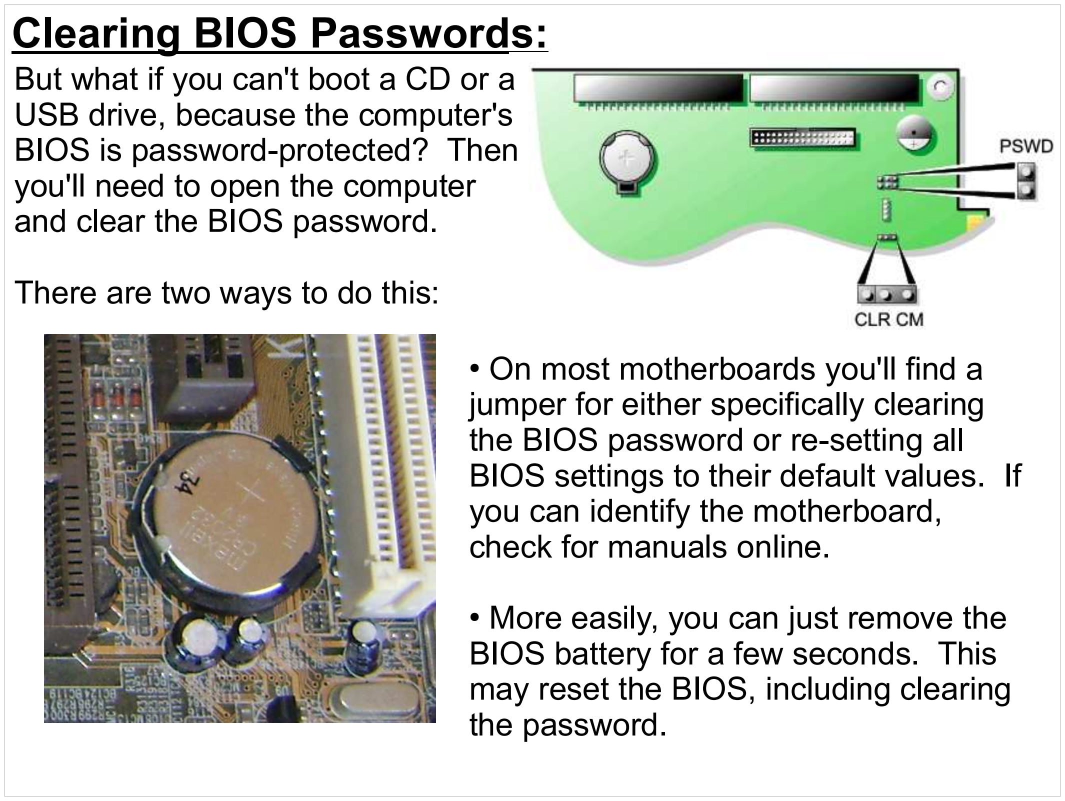
You can do the same using an Ubuntu Live CD, or any of various live distributions stored on a bootable USB drive.

The screenshot above shows a portion of the application menu from KNOPPIX 6.7 (DVD version). Note that the default user under knoppix has no password, so despite what the tooltip says here, no password is required to get a root shell.

With KNOPPIX, you can also just press CTRL-ALT-F1 to get to a different virtual console, already logged in as root. I find this more convenient than using the point-and-click interface.

You can also use the KNOPPIX disk to mount Windows NTFS filesystems and copy files, if necessary. If you need to break into a Windows computer and change passwords, there's another convenient bootable CD called the “Offline NT Registry and Password Editor”:

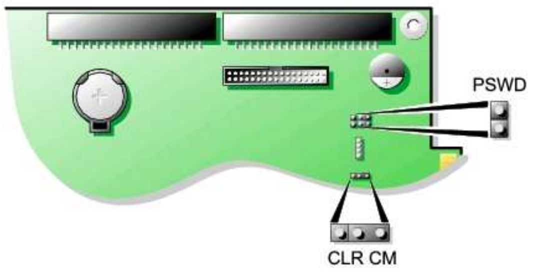
http://pogostick.net/~pnh/ntpasswd/

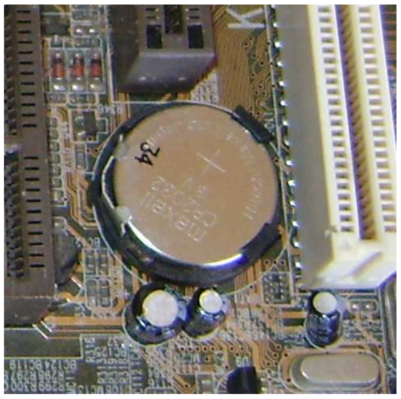


Clearing BIOS Passwords:

But what if you can't boot a CD or a USB drive, because the computer's BIOS is password-protected? Then you'll need to open the computer and clear the BIOS password.

There are two ways to do this:





• On most motherboards you'll find a jumper for either specifically clearing the BIOS password or re-setting all BIOS settings to their default values. If you can identify the motherboard, check for manuals online.

• More easily, you can just remove the BIOS battery for a few seconds. This may reset the BIOS, including clearing the password.

The appropriate jumper will often be labeled "CLR RTC" (realtime clock) or "CLR CMOS" (CMOS is the type of memory used by the BIOS in older computers).

For older computers, removing the battery is often the easiest thing to do. But, it turns out that modern motherboards usually use non-volatile flash memory for storing BIOS settings, and only use the battery to keep the on-board clock running. In these cases, you'll need to locate the appropriate password-clearing jumper. Googling for the motherboard model may turn up useful documentation.

Case Locks:

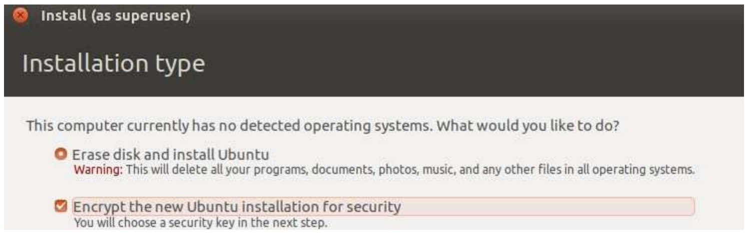
But what if you can't get the case open? Aaaargh! Many cases have built-in locks, and some even provide places to attach padlocks. You can also use special “barrels” that can be padlocked to prevent screws from being removed.

Obviously, any of these can be defeated by a determined person who has physical access to the computer.

Disk Encryption:

But what if someone steals the whole computer? They could break open the case, extract the computer's disk, and use another computer to read your data!

....Unless your disk is encrypted!



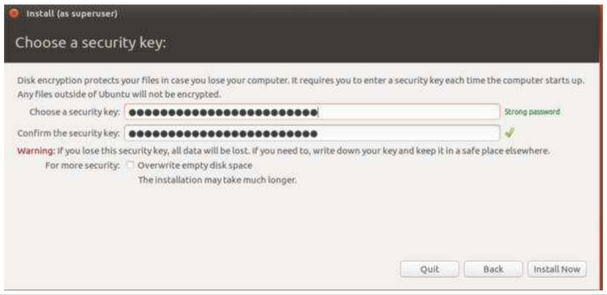
Install (as superuser)

Installation type

This computer currently has no detected operating systems. What would you like to do?

[✓ Erase disk and install Ubuntu]
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

[✓ Encrypt the new Ubuntu installation for security]
You will choose a security key in the next step.



Install (as superuser)

Choose a security key:

Disk encryption protects your files in case you lose your computer. It requires you to enter a security key each time the computer starts up. Any files outside of Ubuntu will not be encrypted.

Choose a security key: ●●●●●●●●●●●●●●●●●●●●●●●● Strong password

Confirm the security key: ●●●●●●●●●●●●●●●●●●●●●●●● ✓

Warning: If you lose this security key, all data will be lost. If you need to, write down your key and keep it in a safe place elsewhere.

For more security: ☐ Overwrite empty disk space
The installation may take much longer.

Most current Linux distributions give you the opportunity to encrypt the disk when you install the operating system. Post-install encryption is more difficult, though. That might require backing up your data, reinstalling the operating system, and restoring the data.

Physical Security:

The device in the upper right is a keylogger. It plugs into the back of your computer, and your keyboard plugs into the keylogger. As you can see, it's quite small, and wouldn't be noticed on the back of a computer. The keylogger logs your keystrokes and saves them on some flash memory. If a Bad Guy puts one of these on your computer, he or she can come by later and pick it up after it captures all of your passwords. (Some models even have WiFi, so the Bad Guy can snoop remotely.) An encrypted disk won't protect you from this, but a locked door might.

We often think that a computer's RAM loses its data as soon as the power is turned off, but the data can actually persist for minutes, or even longer if it's cooled (for example, by spraying a little canned air on it.) This gives a Bad Guy a chance to boot from a thumb drive and read data left in RAM (perhaps your password? or the drive's encryption password?) The data can even persist long enough for the memory module to be moved to a different computer and read there. Again, locked doors can prevent this attack.

Moral:

There is no security without physical security.

This is what security professionals say. The point is that, if someone has physical access to your computer, it's almost certain that they can break into it (and probably with relatively little effort).

Practically, we usually talk about “risk management”. It's never possible to eliminate all risks, but you need to be aware of the risks so you can make intelligent decisions. What's an acceptable level of risk for this particular computer? Think about it, and then try to strike an appropriate balance between usability and security.

Part 3: Checksums, Signatures and Certificates

TRANSFER No. 10440 FOLIO S.3992

CERTIFICATE No 102201

ORDINARY SHARE CERTIFICATE

Rs. 50 SHARES 5

THE Bombay Electric Supply and Tramways Company Limited

INCORPORATED UNDER THE INDIAN COMPANIES ACT 1913.

Capital Rs. 3,90,00,000.

DIVIDED INTO 600,000 ORDINARY SHARES AND 180,000 SEVEN PER CENT CUMULATIVE PREFERENCE SHARES ALL OF 50 RUPEES EACH.

This is to Certify that Shariar Rustom Irani of Bombay is the Registered Holder of Five Ordinary Shares of Fifty Rupees each numbered 545529 to 545533 inclusive in The Bombay Electric Supply and Tramways Company Limited subject to the Memorandum and Articles of Association of the Company and that

The following will be a very high-level, very fast flyover of the topic of certificates. We use digital certificates all the time when we're browsing the web. If you someday set up your own web server, and want to allow encrypted connections to it, you'll need to understand a little bit about what certificates are.

File Checksums:

When we looked at how password information is stored, we saw that a good hash function has the following properties:

• It's deterministic. For any particular input, F(x) will always produce the same output.

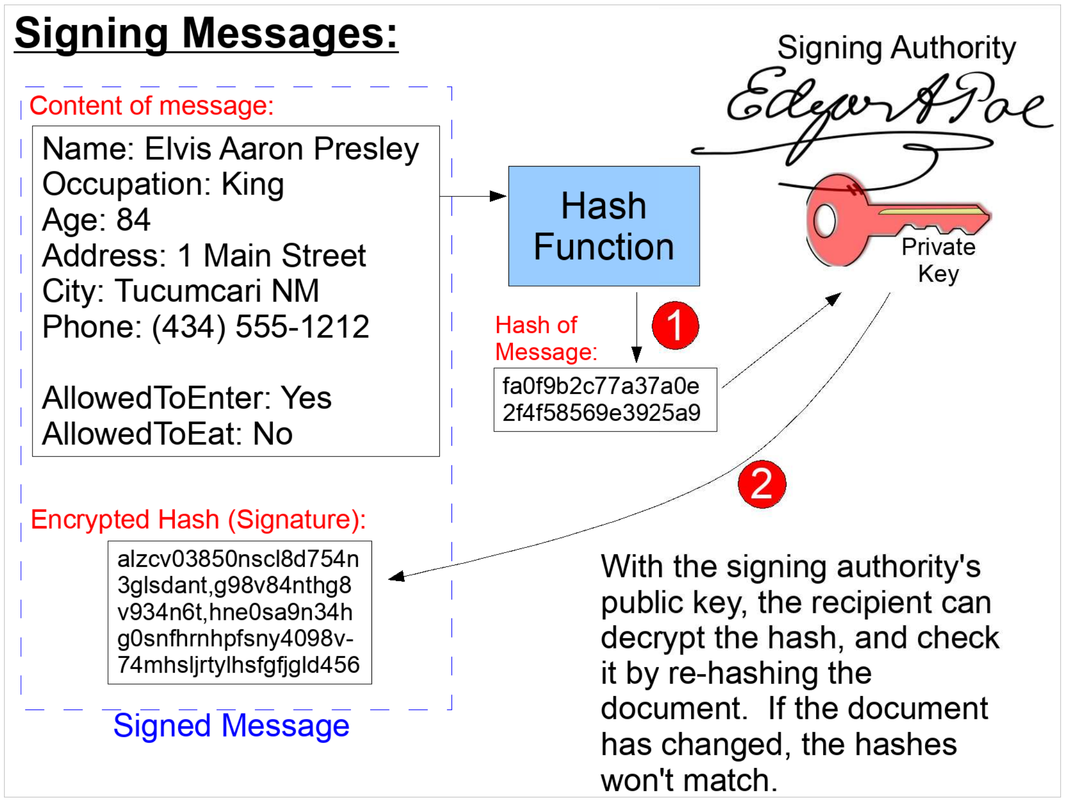
• The output is almost always unique. Different inputs are very unlikely to produce the same output, even if they differ only slightly.

When we run the contents of a file through such a hash function, the resulting hash is called a "checksum". You can generate an MD5 hash of a file by using the "md5sum" command:

```
~/demo> md5sum file.dat
0dd05299d1e021aec53b0a86b791fc92  file.dat
```

You can run this command again, later, to see if the file has changed. If even one byte of the file has changed, the checksum will be different.

At first, it might not seem that this topic is related to certificates, but it's really very important.

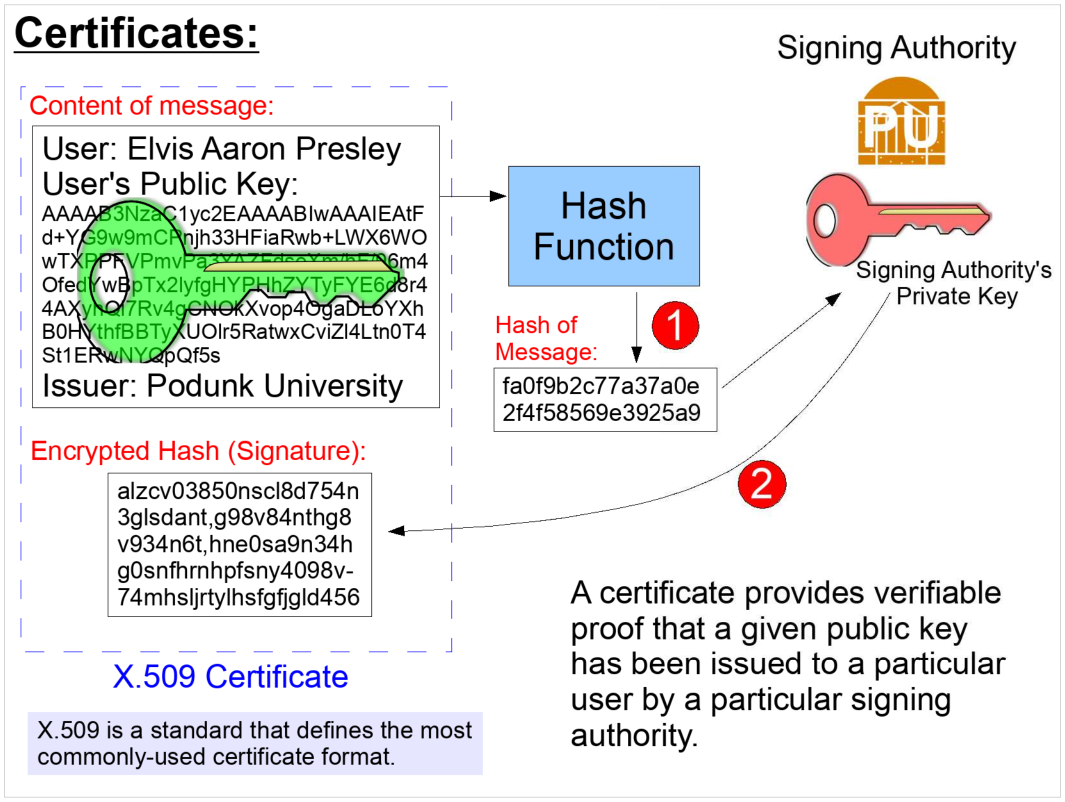


Sometimes we receive an e-mail message with a digital signature attached to it. This shows how those signatures are created using “public key encryption”. This is the kind of encryption we discussed when we talked about SSH. In this scheme, each user has two keys, a public key and a private key.

The signature is just an encrypted checksum of the message content. If the recipient can decrypt the signature by using the sender's public key, then the recipient can be certain that the signature was really created by this particular sender (since, presumably, nobody else has the necessary matching private key.)

Then the recipient can run the message content through a hash function, and check to see if the resulting checksum matches the encrypted checksum that was created by the sender. If they match, the recipient can be sure that the message hasn't been altered.

Many mail systems provide you with tools that can automatically check signatures and create signed message.



User: Elvis Aaron Presley
User's Public Key:
AAAAB3NzaC1yc2EAAAABIwAAAIEAtF
d+YG9w9mCPnjh33HFiaRwb+LWX6WO
wTXPPFVPmvPa3YAZEdsoYm/bF/06m4
OfedYwBpTx2lyfgHYPHhZYTyFYE6d8r4
4AXynQl7Rv4gCNOkXvop4OgaDLoYXh
B0HYthfBBTyXUOlr5RatwxCviZl4Ltn0T4
St1ERwNYQpQf5s
Issuer: Podunk University

fa0f9b2c77a37a0e
2f4f58569e3925a9

alzcv03850nscl8d754n
3glsdant,g98v84nthg8
v934n6t,hne0sa9n34h
g0snfhrnhpfsny4098v-
74mhsljrtylhsfgfjgld456

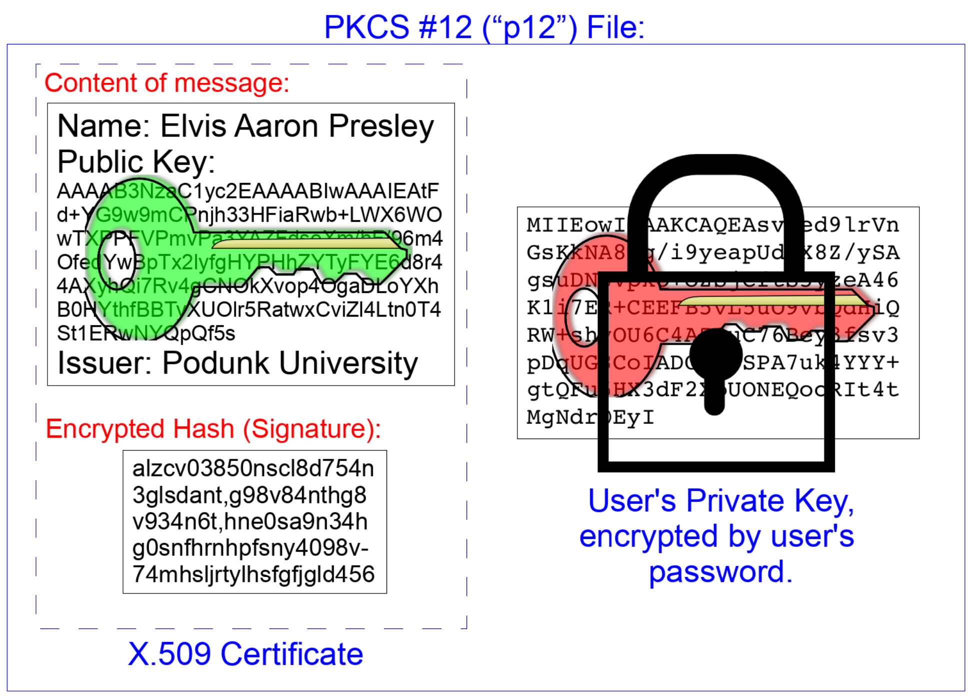
X.509 is a standard that defines the most commonly-used certificate format.

A certificate provides verifiable proof that a given public key has been issued to a particular user by a particular signing authority.

A digital certificate is a particular type of signed message whose content is a user's public key. The signing authority has assigned this key to the user, and the certificate provides verifiable proof of this.

X.509 is the certificate format used by web servers and web browsers.

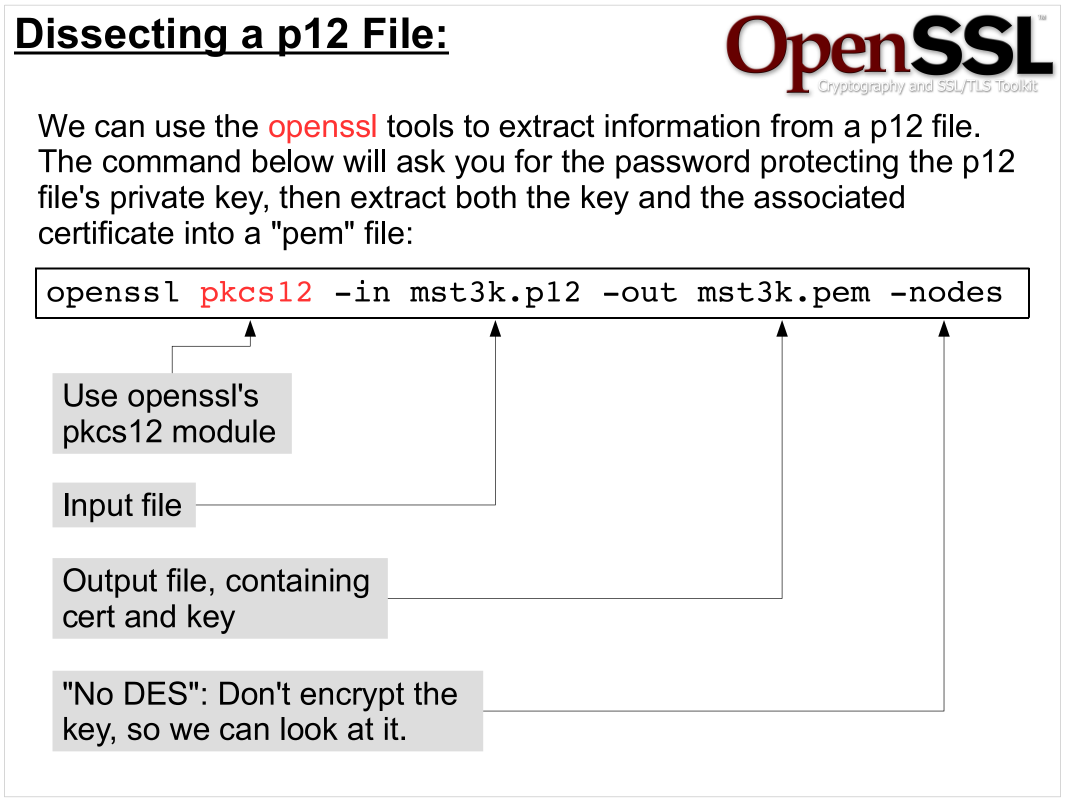
Certificate Distribution:



When an authority provides you with a public key, they also need to provide you with the matching private key. But how can they securely deliver the private key to you?

Typically, the public key (inside its certificate) is packed up together with an encrypted version of the private key in a thing called a PKCS #12 file (for "Public Key Certificate Standard Number 12"). The private key is encrypted through a symmetric encryption process, using a password provided by you as the key.

Once you receive this file, you can load it into your web browser, and the browser can then use this information to authenticate you to web services.



Dissecting a p12 File:

We can use the openssl tools to extract information from a p12 file. The command below will ask you for the password protecting the p12 file's private key, then extract both the key and the associated certificate into a "pem" file:

```
openssl pkcs12 -in mst3k.p12 -out mst3k.pem -nodes
```

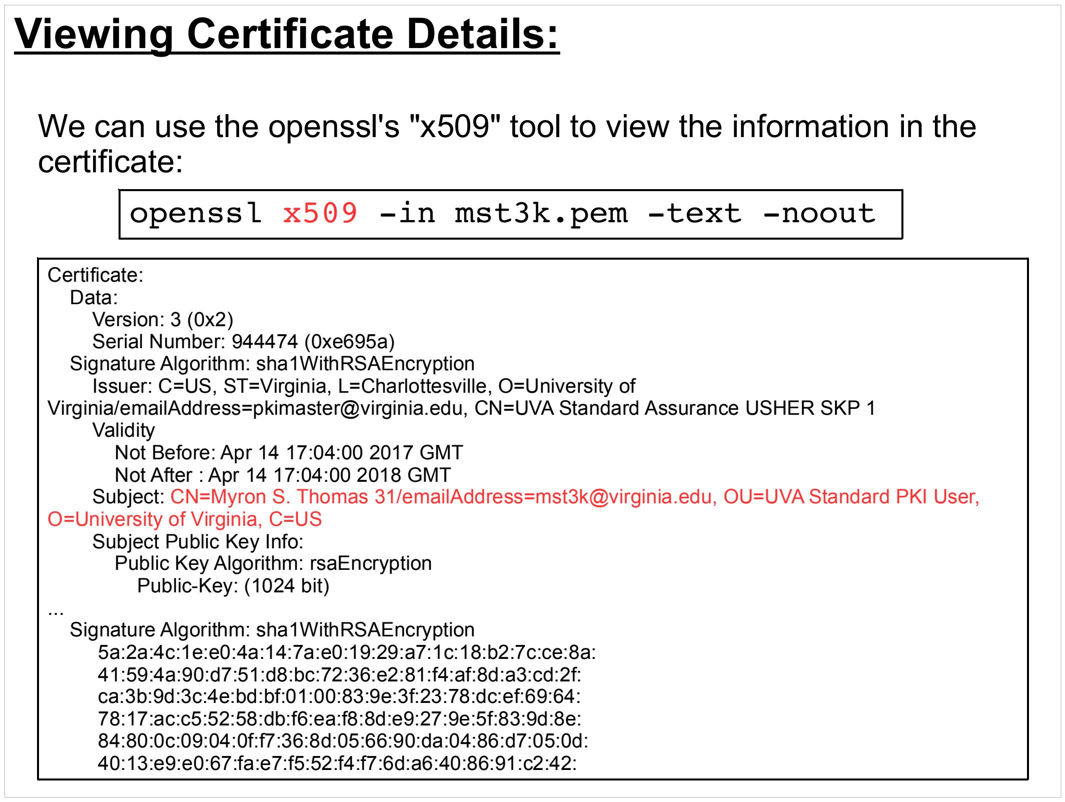
Use openssl's pkcs12 module

Input file

Output file, containing cert and key

"No DES": Don't encrypt the key, so we can look at it.

The openssl command has several "sub-commands" including "pkcs12", "x509", "req", and "genrsa". For a full list, see the openssl man page.



Viewing Certificate Details:

We can use the openssl's "x509" tool to view the information in the certificate:

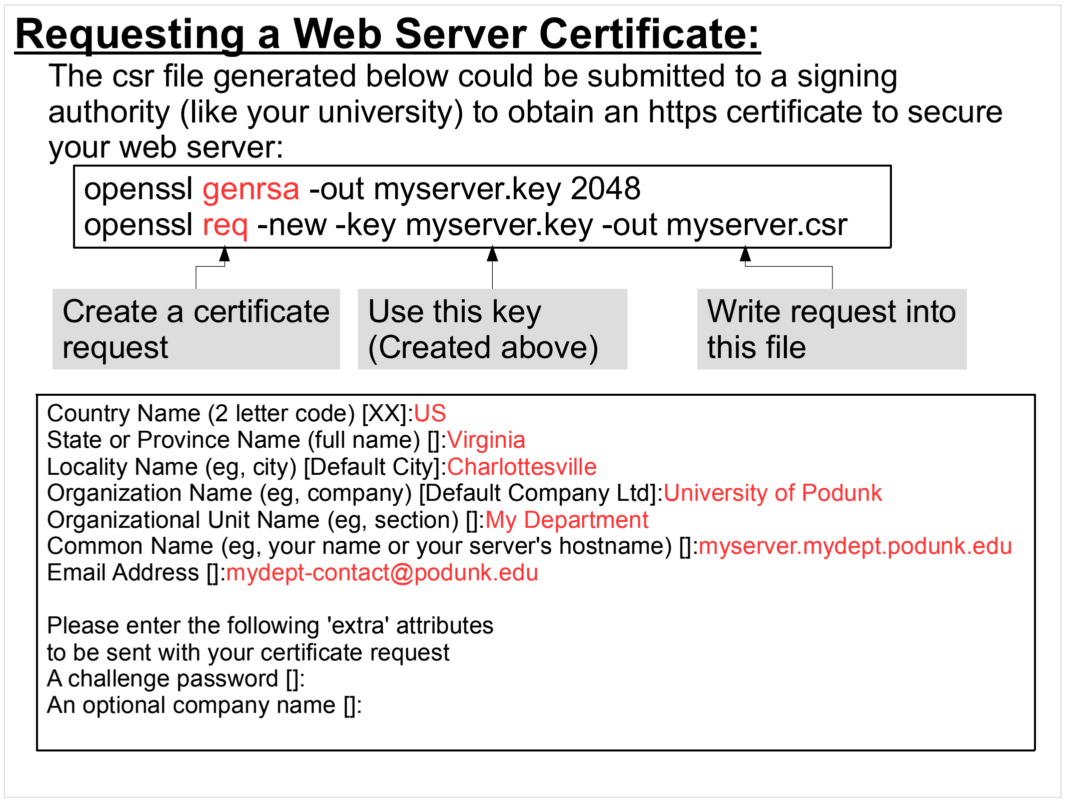
```
openssl x509 -in mst3k.pem -text -noout
```

```
Certificate:
   Data:
      Version: 3 (0x2)
      Serial Number: 944474 (0xe695a)
   Signature Algorithm: sha1WithRSAEncryption
      Issuer: C=US, ST=Virginia, L=Charlottesville, O=University of
Virginia/emailAddress=pkimaster@virginia.edu, CN=UVA Standard Assurance USHER SKP 1
      Validity
         Not Before: Apr 14 17:04:00 2017 GMT
         Not After : Apr 14 17:04:00 2018 GMT
      Subject: CN=Myron S. Thomas 31/emailAddress=mst3k@virginia.edu, OU=UVA Standard PKI User,
O=University of Virginia, C=US
      Subject Public Key Info:
         Public Key Algorithm: rsaEncryption
            Public-Key: (1024 bit)
...
   Signature Algorithm: sha1WithRSAEncryption
       5a:2a:4c:1e:e0:4a:14:7a:e0:19:29:a7:1c:18:b2:7c:ce:8a:
       41:59:4a:90:d7:51:d8:bc:72:36:e2:81:f4:af:8d:a3:cd:2f:
       ca:3b:9d:3c:4e:bd:bf:01:00:83:9e:3f:23:78:dc:ef:69:64:
       78:17:ac:c5:52:58:db:f6:ea:f8:8d:e9:27:9e:5f:83:9d:8e:
       84:80:0c:09:04:0f:f7:36:8d:05:66:90:da:04:86:d7:05:0d:
       40:13:e9:e0:67:fa:e7:f5:52:f4:f7:6d:a6:40:86:91:c2:42:
```

As we mentioned before, X.509 is the standard certificate format used by web servers and web browsers. Openssl's x509 module has the ability to read and manipulate these certificates.

In the example output above, I've omitted the public key and a lot of other stuff. There's quite a bit of data packed into one of these certificates.

Notice that the certificate specifies that it's only valid during a certain time period (April 2017 through April 2018).



Requesting a Web Server Certificate:

The csr file generated below could be submitted to a signing authority (like your university) to obtain an https certificate to secure your web server:

```
openssl genrsa -out myserver.key 2048
openssl req -new -key myserver.key -out myserver.csr
```

Create a certificate request
Use this key (Created above)
Write request into this file

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:Virginia
Locality Name (eg, city) [Default City]:Charlottesville
Organization Name (eg, company) [Default Company Ltd]:University of Podunk
Organizational Unit Name (eg, section) []:My Department
Common Name (eg, your name or your server's hostname) []:myserver.mydept.podunk.edu
Email Address []:mydept-contact@podunk.edu

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

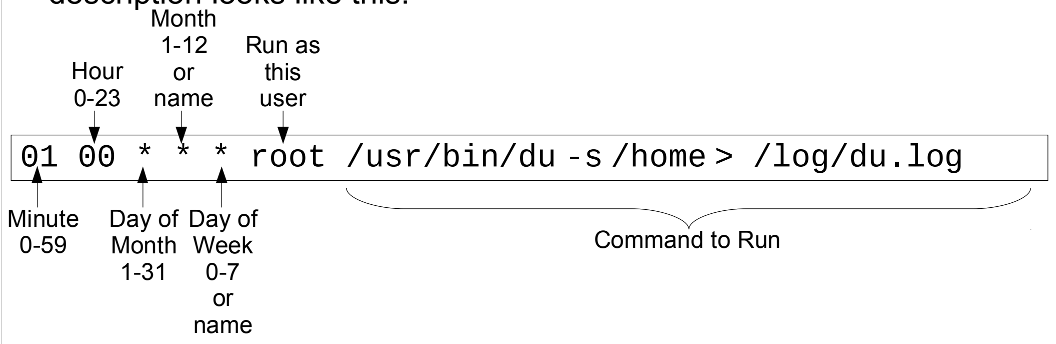
The csr file generated by this process can be sent to a local signing authority. The signing authority will then send you back a signed certificate that you can use with your web server to encrypt https connections and confirm that your server really is what it says it is.

Part 4: Miscellaneous Topics

These are several short topics that we couldn't fit in earlier.

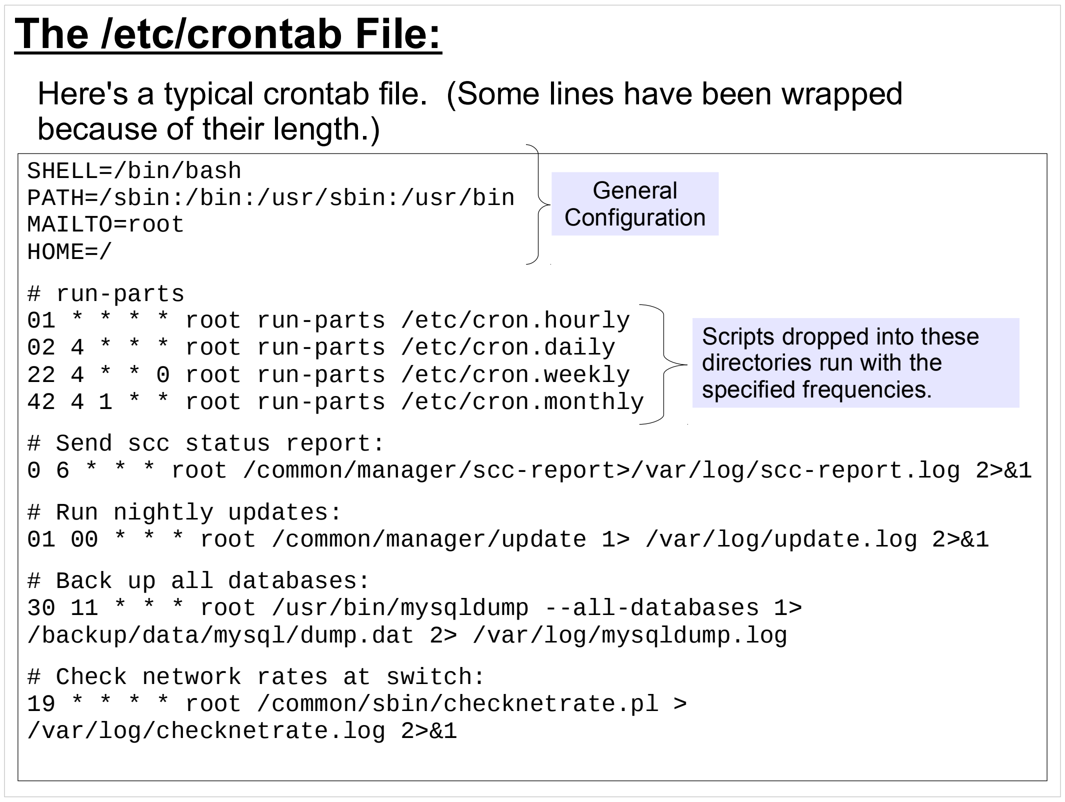
The cron Service:

Cron is a service that can periodically, automatically, execute a given command. It decides what to do based on a system-wide "crontab" file and the crontab files (if any) of individual users. Each periodic job is defined by a line in one of these files. A job description looks like this:



```
01 00 * * * root /usr/bin/du -s /home > /log/du.log
```

• For months, you can use either numbers or "Jan", "Feb", etc.
• For days of the week, both 0 and 7 designate Sunday, or you can use "Sun", "Mon", etc.
• "*" means every day, every month, etc.
• Cron checks for jobs to do once per minute, so a "*" in all of the first five fields will cause cron to execute the job once per minute.



The /etc/crontab File:

Here's a typical crontab file. (Some lines have been wrapped because of their length.)

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly

# Send scc status report:
0 6 * * * root /common/manager/scc-report>/var/log/scc-report.log 2>&1

# Run nightly updates:
01 00 * * * root /common/manager/update 1> /var/log/update.log 2>&1

# Back up all databases:
30 11 * * * root /usr/bin/mysqldump --all-databases 1>
/backup/data/mysql/dump.dat 2> /var/log/mysqldump.log

# Check network rates at switch:
19 * * * * root /common/sbin/checknetrate.pl >
/var/log/checknetrate.log 2>&1
```

General Configuration

Scripts dropped into these directories run with the specified frequencies.

If a command doesn't explicitly redirect its output, any output is e-mailed to the address specified in the “MAILTO” configuration parameter (root, by default).

The “crontab” Command, and Personal Crontabs:

Crontab files for individual users can be created and managed through the “crontab” command.

The following command will let you edit your own crontab file, or create it if it doesn't exist:

```
~/demo> crontab -e
```

The crontab command above will invoke your preferred text editor (as defined by the setting of the “VISUAL” or “EDITOR” environment variable). Job entries are in the same format as the system-wide /etc/crontab file, except that the username field is omitted.

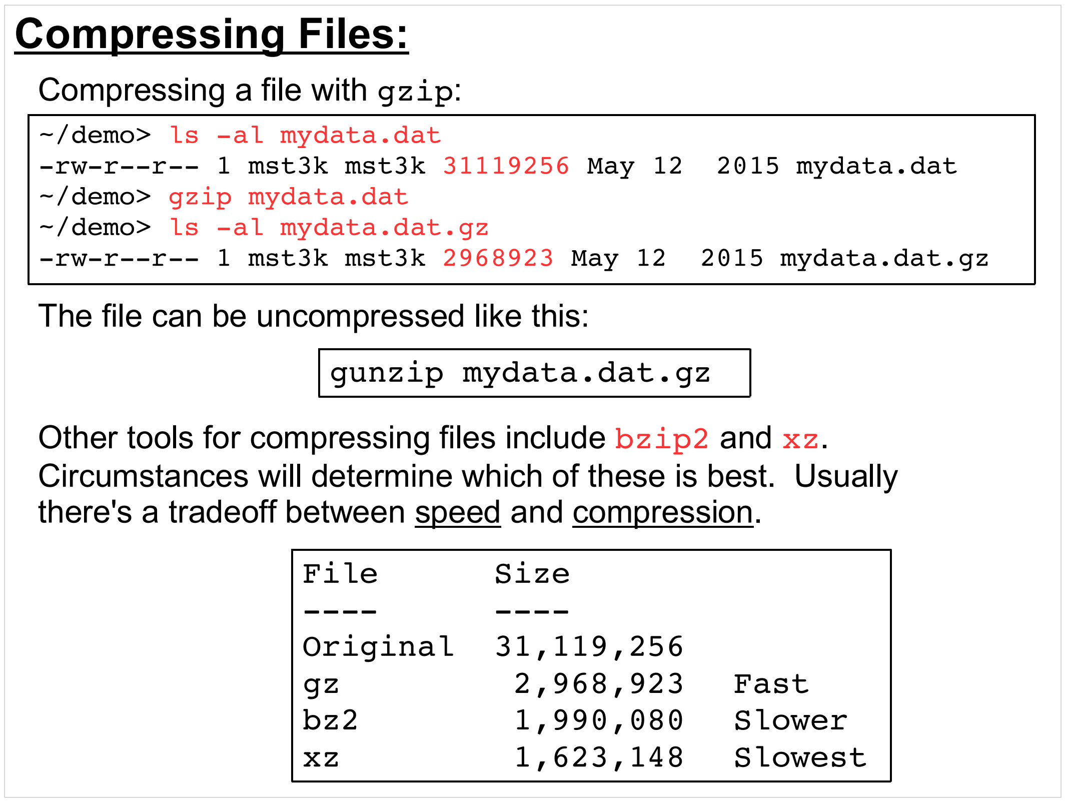
This will remove your existing crontab (without asking for confirmation):

```
~/demo> crontab -r
```

This will show you the contents of your crontab, if any:

```
~/demo> crontab -l
00 12 * * * ps auxwwwf
```

The crontab files for individual users typically live in the directory /var/spool/cron.



Compressing Files:

Compressing a file with `gzip`:

```
~/demo> ls -al mydata.dat
-rw-r--r-- 1 mst3k mst3k 31119256 May 12  2015 mydata.dat
~/demo> gzip mydata.dat
~/demo> ls -al mydata.dat.gz
-rw-r--r-- 1 mst3k mst3k 2968923 May 12  2015 mydata.dat.gz
```

The file can be uncompressed like this:

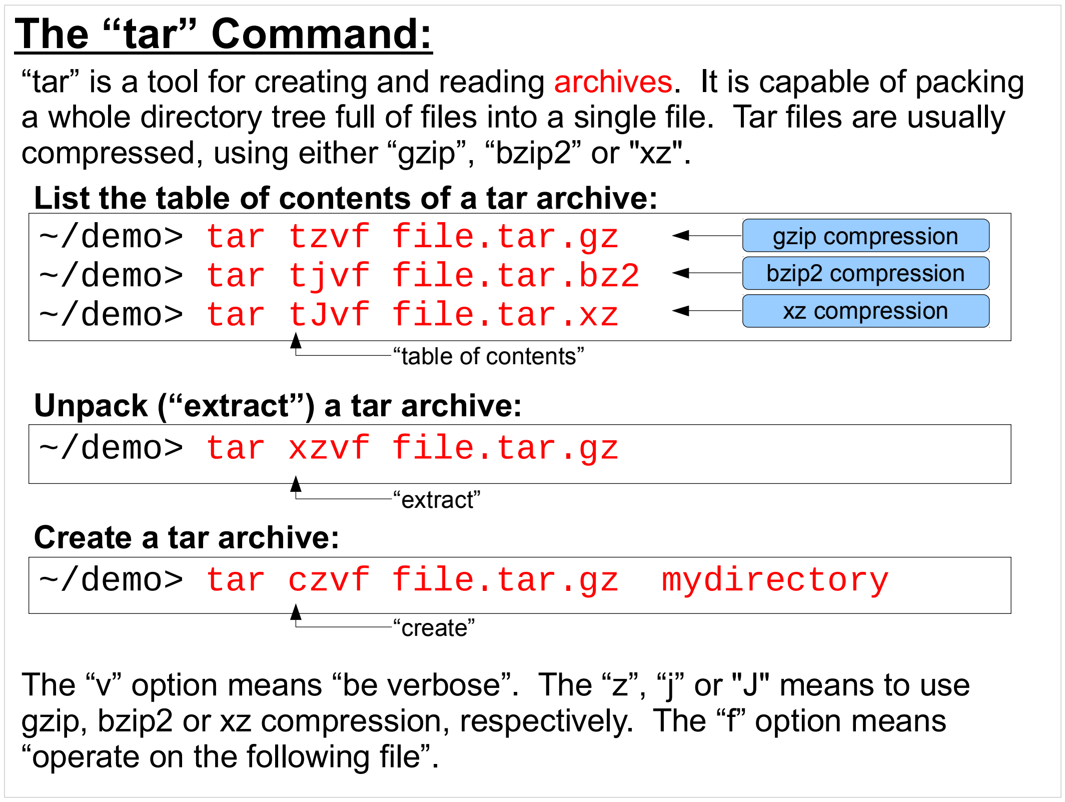
```
gunzip mydata.dat.gz
```

Other tools for compressing files include `bzip2` and `xz`. Circumstances will determine which of these is best. Usually there's a tradeoff between speed and compression.

File	Size	
----	----	
Original	31,119,256	
gz	2,968,923	Fast
bz2	1,990,080	Slower
xz	1,623,148	Slowest

The speed of these compression tools isn't particularly relevant when we're only compressing or decompressing one file. In that case it doesn't matter whether the process takes one second or twenty seconds. But if we're dealing with thousands of files, we need to think about the speed.

Files vary greatly in their compressibility. Text files (anything you can edit with a text editor) tend to be highly compressible. Most image file formats (jpeg, png, gif, etc.) are already compressed, so using gzip on them won't decrease their size (and might even increase it a little).



The “tar” Command:

“tar” is a tool for creating and reading archives. It is capable of packing a whole directory tree full of files into a single file. Tar files are usually compressed, using either “gzip”, “bzip2” or "xz".

List the table of contents of a tar archive:

```
~/demo> tar tzvf file.tar.gz
~/demo> tjvf file.tar.bz2
~/demo> tar tJvf file.tar.xz
```

gzip compression
bzip2 compression
xz compression
“table of contents”

Unpack (“extract”) a tar archive:

```
~/demo> tar xzvf file.tar.gz
```

Create a tar archive:

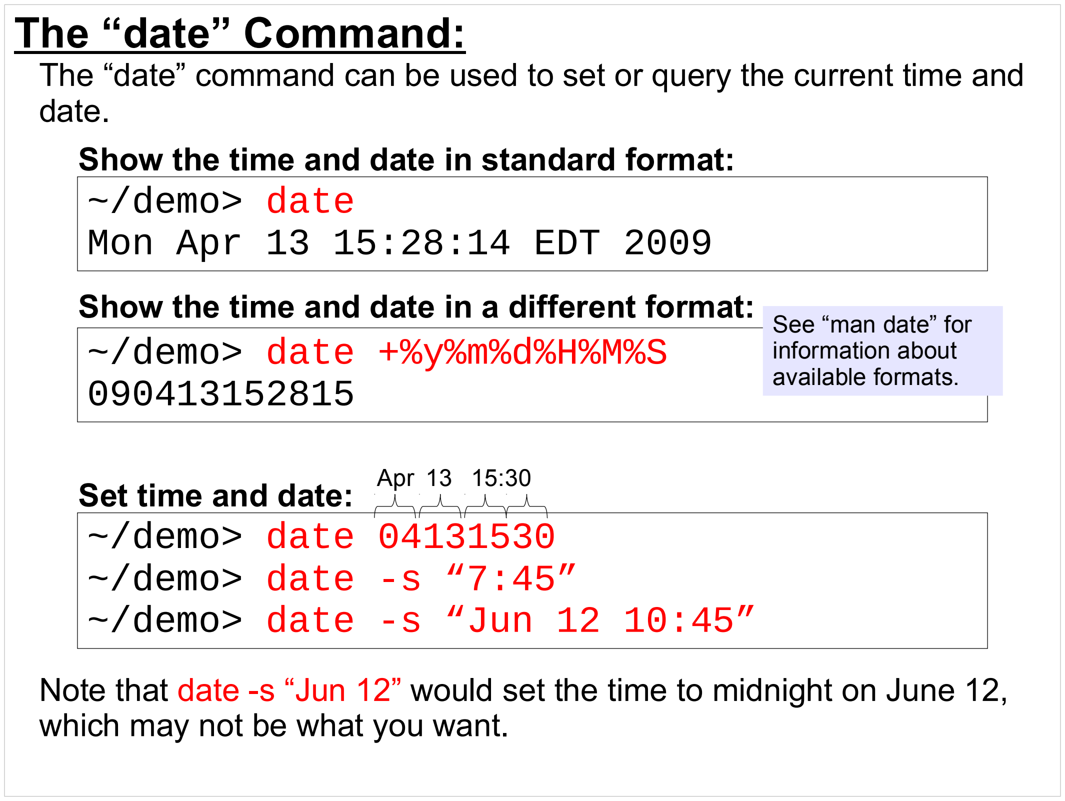
```
~/demo> tar czvf file.tar.gz  mydirectory
```

The “v” option means “be verbose”. The “z”, “j” or "J" means to use gzip, bzip2 or xz compression, respectively. The “f” option means “operate on the following file”.

The name “tar” comes from “tape archive”, which is what tar was originally used for. You could, for example, type:

tar czvf /dev/rmt0 /home

To back up the directory “/home” onto a magnetic tape device called “/dev/rmt0”.



The “date” Command:

The “date” command can be used to set or query the current time and date.

Show the time and date in standard format:

```
~/demo> date
Mon Apr 13 15:28:14 EDT 2009
```

Show the time and date in a different format:

```
~/demo> date +%y%m%d%H%M%S
090413152815
```

See “man date” for information about available formats.

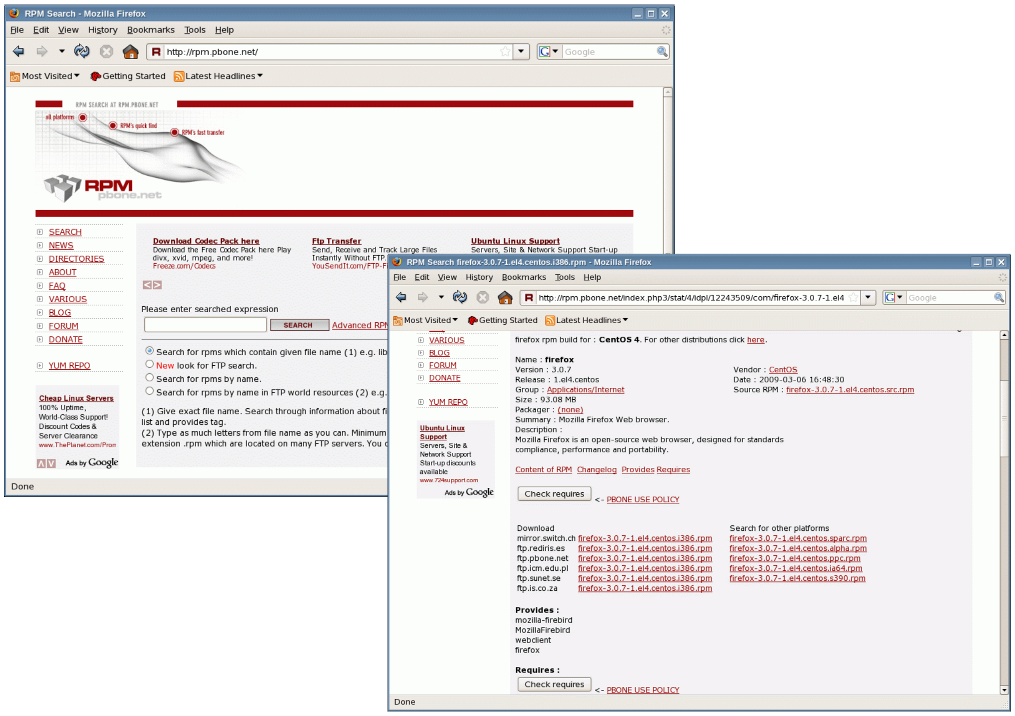
Set time and date:

```
                 Apr 13  15:30
~/demo> date 04131530
~/demo> date -s “7:45”
~/demo> date -s “Jun 12 10:45”
```

Note that date -s “Jun 12” would set the time to midnight on June 12, which may not be what you want.

The “date +%y%m%d%H%M%S” format is useful in scripts, since the output makes a handy file or directory name.

Finding RPM packages using rpm.pbone.net:


RPM Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://rpm.pbone.net/
Most Visited Getting Started Latest Headlines
all platforms
RPM's quick find
RPM's fast transfer
RPM pbone.net
SEARCH
NEWS
DIRECTORIES
ABOUT
FAQ
VARIOUS
BLOG
FORUM
DONATE
YUM REPO
SEARCH
Done
RPM Search firefox-3.0.7-1.el4.centos.i386.rpm - Mozilla Firefox
http://rpm.pbone.net/index.php3/stat/4/idpl/12243509/com/firefox-3.0.7-1.el4
Check requires
Done

Please enter searched expression

Search for rpms which contain given file name (1) e.g. lib

New look for FTP search.

Search for rpms by name.

Search for rpms by name in FTP world resources (2) e.g.

(1) Give exact file name. Search through information about fi
list and provides tag.
(2) Type as much letters from file name as you can. Minimum
extension .rpm which are located on many FTP servers. You c

firefox rpm build for : **CentOS 4**. For other distributions click here.

```
Name : firefox
Version : 3.0.7                     Vendor : CentOS
Release : 1.el4.centos              Date : 2009-03-06 16:48:30
Group : Applications/Internet       Source RPM : firefox-3.0.7-1.el4.centos.src.rpm
Size : 93.08 MB
Packager : (none)
Summary : Mozilla Firefox Web browser.
Description :
Mozilla Firefox is an open-source web browser, designed for standards
compliance, performance and portability.

Content of RPM  Changelog  Provides Requires

Download                                                  Search for other platforms
mirror.switch.ch firefox-3.0.7-1.el4.centos.i386.rpm      firefox-3.0.7-1.el4.centos.sparc.rpm
ftp.rediris.es   firefox-3.0.7-1.el4.centos.i386.rpm      firefox-3.0.7-1.el4.centos.alpha.rpm
ftp.pbone.net    firefox-3.0.7-1.el4.centos.i386.rpm      firefox-3.0.7-1.el4.centos.ppc.rpm
ftp.icm.edu.pl   firefox-3.0.7-1.el4.centos.i386.rpm      firefox-3.0.7-1.el4.centos.ia64.rpm
ftp.sunet.se     firefox-3.0.7-1.el4.centos.i386.rpm      firefox-3.0.7-1.el4.centos.s390.rpm
ftp.is.co.za     firefox-3.0.7-1.el4.centos.i386.rpm

Provides :
mozilla-firebird
MozillaFirebird
webclient
firefox

Requires :
```

<- PBONE USE POLICY

We all use Google regularly, but there are a few other really useful search engines. This is one of them.

If you're having trouble finding an RPM package, try rpm.pbone.net. It's an immense, searchable index of RPM repositories.

If you're looking for debian packages, take a look at:

http://packages.debian.org

The “limit” and “ulimit” Commands:

Shells typically provide a way to set limits on the CPU time, memory, and other resources that child processes can use. For example:

Viewing limits under tcsh:

```
~/demo> limit
cputime        unlimited
filesize       unlimited
datasize       unlimited
stacksize      10240 kbytes
coredumpsize   0 kbytes
memoryuse      unlimited
vmemoryuse     unlimited
descriptors    1024
memorylocked   unlimited
maxproc        30428
```

Viewing limits under bash:

```
~/demo> ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 29
file size               (blocks, -f) unlimited
pending signals                 (-i) 30428
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 99
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 30428
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

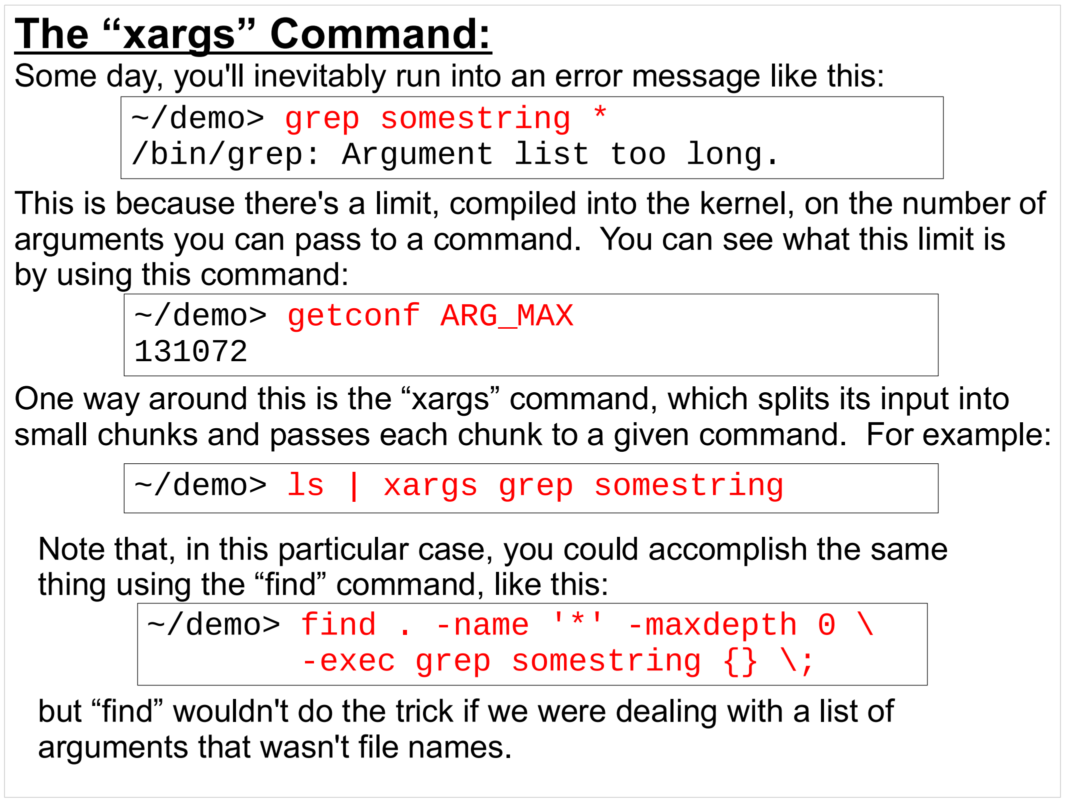
Setting limits under tcsh:

```
~/demo> limit coredumpsize 0
```

Setting limits under bash:

```
~/demo> ulimit -c 0
```

Usually, initial limits for all users are set in the /etc/csh.cshrc or /etc/profile files. For both of these shells, limits may be “hard” or “soft”. After a “hard” limit is set, it cannot thereafter be increased, except by the root user.



The “xargs” Command:

Some day, you'll inevitably run into an error message like this:

```
~/demo> grep somestring *
/bin/grep: Argument list too long.
```

This is because there's a limit, compiled into the kernel, on the number of arguments you can pass to a command. You can see what this limit is by using this command:

```
~/demo> getconf ARG_MAX
131072
```

One way around this is the “xargs” command, which splits its input into small chunks and passes each chunk to a given command. For example:

```
~/demo> ls | xargs grep somestring
```

Note that, in this particular case, you could accomplish the same thing using the “find” command, like this:

```
~/demo> find . -name '*' -maxdepth 0 \
        -exec grep somestring {} \;
```

but “find” wouldn't do the trick if we were dealing with a list of arguments that wasn't file names.

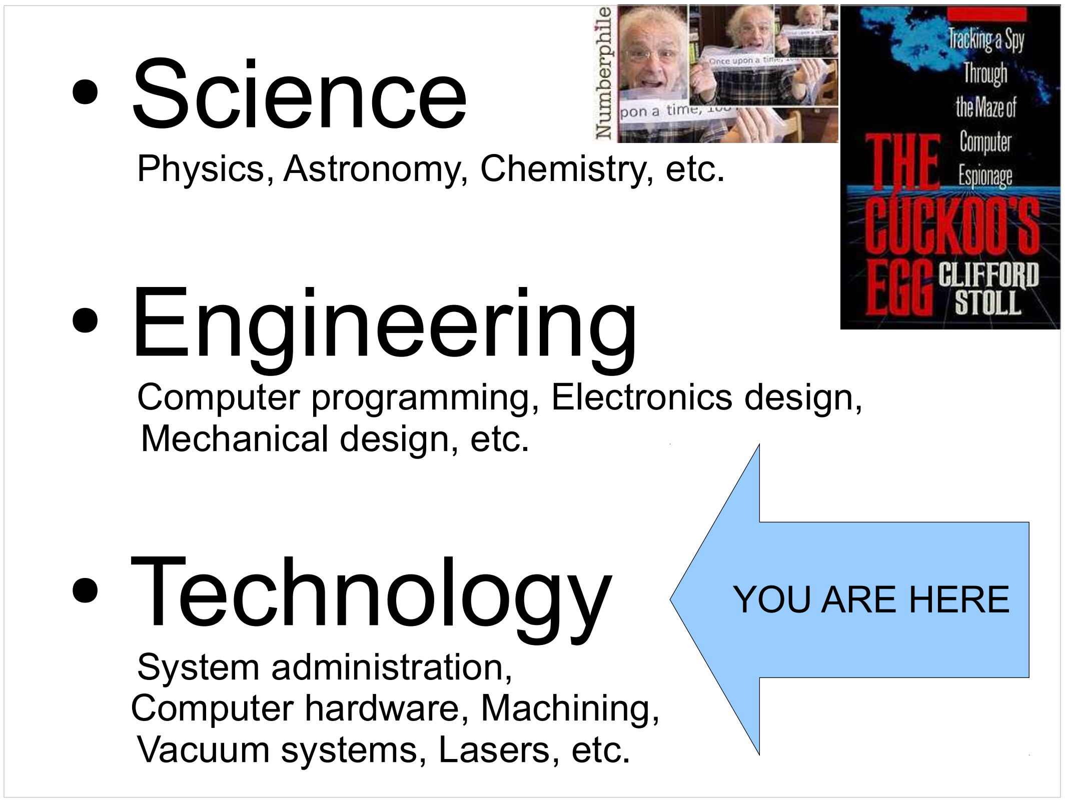
...also, the “find” command above is much longer, and thus more prone to typos.

Part 4: What's It All About?

So, now that we're at the end of this course, what was the purpose of it?

Let's transform to a different coordinate system for a minute, an pretend we're all Architects instead of Physicists. In that realm we might say that this course didn't teach you how to design a house, and it didn't even teach you how to build a house. What it taught (or tried to) was simply how to operate a hammer.

The tools we've been looking at are some of the hammers, saws and dangerous power tools that you may find useful at some point in your career.



- Science

  Physics, Astronomy, Chemistry, etc.

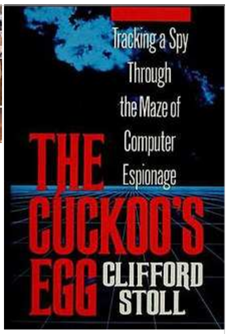
- Engineering

  Computer programming, Electronics design, Mechanical design, etc.

- Technology

  System administration, Computer hardware, Machining, Vacuum systems, Lasers, etc.

YOU ARE HERE



A while back I saw a talk by astronomer Cliff Stoll, in which he said he wasn't interested in computers any more because they aren't science and they aren't even engineering. These days, they're just “technology”. Whether you share his sentiments or not, I think he's given us a useful way of categorizing things.

While we're in school, we get a very good education in Science, and we get a fair amount of formal training in engineering: learning to write programs for data analysis or simulations, learning to design electronics, and so forth.

We get less training in “technology”, though, and that's what I hoped this course would help provide.

I hope you feel that your toolbox is a little fuller after this, and that you've picked up at least a few things that will help you do good science (and engineering) later on.

Thanks!