Linux and Unix

- Specify section number with Unix man (manual) command
- Bash scripting
- lsof, the least talked about Unix tool
- In Unix, what do I do when a file I want to edit in vi says "Line too long"?
- How can I check/verify that Red Hat Enterprise Linux is up-to-date for Daylight Savings Time
- 15 Basic ‘ls’ Command Examples in Linux
- cron job examples
- adding date to bash history command
- How do I extract certain columns from a text file in Unix?
- Command Line Tools to Monitor Linux Performance
- Awk Explained
- Finding most recently changed files in Unix
- Thin Client technology - Linux Terminal Server Project HOWTO
- combining unix stderr output with stdout
- How do I keep color output when paginating shell output through less?
- Unix Cheat Sheets and Tricks
- vi editing notes
- Linux scp command examples
- diff and patch
- UNIX Tutorials
- Linux tee command
- Manipulating data on Linux
- Advantages of less over more (UNIX)
- Case-insensitive searching in vi
- Customizing vi
- PHP Client can break `less`
- How to synchronize files and directories from one location to another on Unix
- avoiding increasing indents while pasting into vi
- SAS on 64b Ubuntu Linux

- Ruby one-liner to count logins in Linux last command
- Linux performance analysis in 60 seconds
- Reasons to Use Vim over Vi
- Unix grep, find and maxdepth
- Useful Commands for Vi
- using find and chmod together in Unix
- finding words in binary files
- grep tricks
- Location of the sudo log file

Specify section number with Unix man (manual) command

Sometimes, there are multiple man pages under the same name, identified by section numbers. For example, there is a MKDIR user command and a MKDIR programming function call. if you just type “man mkdir” at the prompt, it might take you to one but not the other.

You can add a -s flag to specify the section number , e.g.

- man -S 1 mkdir # shows the manual for MKDIR
- man -S 2 mkdir # shows the manual for MKDIR

Bash scripting

- Learning Shell Scripting Language: A Guide from Newbies to System Administrator – October 3, 2013
- Bash shell scripting Tutorial – Part I
- Advanced Bash-Scripting Guide
- bash Cookbook – UCLA only – licensed through O'Reilly Safari books online
- Unix: Debugging your scripts
- 10 Useful Chaining Operators in Linux with Practical Examples – December 13, 2013
- Bash Scripting Quirks & Safety Tips – *March 26, 2017* Includes links to other resources, e.g. Command Line Challenge Game
- Advanced Bash-Scripting Guide: Manipulating Strings

Please add Bash links that you've found useful.

lsof, the least talked about Unix tool

Apparently this is one of the least talked about, but most useful Linux/Unix tools.

lsof lists information about files opened by processes, and according to these articles, that is more useful than it sounds.

- http://dmiessler.com/study/lsof/
- http://0xfe.blogspot.com/2006/03/troubleshooting-unix-systems-with-lsof.html

Heard about this on http://reddit.com

In Unix, what do I do when a file I want to edit in vi says "Line too long"?

In Unix, when the vi editor says “line too long” it’s usually because it was copied up from a Mac where the end of line marker is different.

This command works for me on Solaris:

```
tr '\015'   '\012' < in_file > out_file
```

Source: http://www.maths.ox.ac.uk/help/faqs/files/dos-mac-to-unix.shtml

The numbers are in octal. Be sure and transpose from decimal or hex if necessary.

- ASCII Character Codes Conversion Chart

How can I check/verify that Red Hat Enterprise Linux is up-to-date for Daylight Savings Time

As the root user, enter the following at the command line:

$ zdump -v /etc/localtime | grep 2007

You should see Mar 11 as one of the dates, if not your system needs to be updated. For more info, go here:

http://kbase.redhat.com/faq/FAQ_80_7909.shtm

15 Basic ‘ls’ Command Examples in Linux

I learned at least three things from this list, and I’ve been using *ls* for years.

15 Basic ‘ls’ Command Examples in Linux

The new things for me were:

- *ls -lh* – human readable format on file sizes
- *ls -ltr* – reverse output order, which is particularly useful when sorting by date
- *ls -lS* – sort by file size
- *ls -R* – Recursively list Sub-Directories (can be combined with S for size)

Note: so far, all of these commands work for me in OS/X as well.

cron job examples

Here are some interesting examples of cron job use.

- http://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/ – *I didn't know about @monthly, @yearly, and @reboot, or changing the MAIL destination setting.*
- http://www.pantz.org/software/cron/croninfo.html
- http://www.adminschoice.com/crontab-quick-reference

== updated by Shinn on 2/22/2013 =====

*/(number) only worked in Linux, not Solaris, so were @yearly, @monthly and @daily.

MAILTO="someone@somedomain" can be used in CentOS to redirect cron output to other account.

adding date to bash history command

Coming from Solaris to Linux, one thing I've missed for years was that the Linux bash *history* command doesn't show timestamps. Thanks to my colleague, Shinn, I learned about this config settings that can go into .bashrc to solve that problem, and add a few other useful settings, as well. In Debian, the (system) file to modify is /etc/bash.bashrc, and /etc/bashrc in CentOS. Of course, user can modify her/his .bashrc.

`export HISTTIMEFORMAT='%F %T '`

Before:

```
  355  vi .bashrc
  356  history
  357  history |  wc -l
```

After:

```
  355  2013-03-15 10:07:03 vi .bashrc
  356  2013-03-15 10:07:17 history
  357  2013-03-15 10:07:48 history |  wc -l
  358  2013-03-15 10:13:43 cat .bashrc
  359  2013-03-15 10:57:46 exit
  360  2013-03-15 13:22:12 history
  361  2013-03-15 13:22:35 more .bashrc
```

`export HISTSIZE=1000` – extends history from the default 500 most recent commands, to 1000.

`export HISTCONTROL=ignoredups` – removes duplicates from history

`export HISTCONTROL=ignoreboth` – I'm not sure about this one.

Please add other config options you find useful.

How do I extract certain columns from a text file in Unix?

Somehow I can never remember the *cut* command in Unix. But I occasionally want to remove certain columns from a text file of data. *cut* will do that.

Delimited

Here is a simple tab-delimited example. (Use the -d option to set a different column delimiter.)

Data File (tab-delimited)

```
111     222     3333    444     555     666
111     222     3333    444     555     666
111     222     3333    444     555     666
111     222     3333    444     555     666
111     222     3333    444     555     666
```

Command

```
cut -f1,4,6 filename
```

Output

```
111     444     666
111     444     666
111     444     666
111     444     666
```

```
111     444     666
```

There are two other examples in the Search and Replace section of this KB article.

https://kb.ucla.edu/link/117

More options are available in the Unix man pages for cut. Type *man cut*

Fixed Width

Use the *-c* option to select characters based on position.

Data File (fixed width)

```
111     222     3333    444     555     666
111     222     3333    444     555     666
111     222     3333    444     555     666
111     222     3333    444     555     666
111     222     3333    444     555     666
```

Command

```
cut -c9,16 filename
```

Output

```
222
222
222
222
222
```

Command Line Tools to Monitor Linux Performance

If you have any tools for monitoring Linux, or have comments or limitations on these, please add them to this article. Thanks,

- 15 Command Line Tools to Monitor Linux Performance

Awk Explained

Awk is a Unix tool that can be very useful for dealing with text but can be hard to understand. Here are some useful explanations.

- awk is a beautiful tool
- Famous Awk One-Liners Explained, Part I
- 8 Powerful Awk Built-in Variables – FS, OFS, RS, ORS, NR, NF, FILENAME, FNR
- How to use awk command in Linux
- Handy One-Line Scripts for Awk – Eric Pement

Finding most recently changed files in Unix

How do I find the most recently changed files in a set of subdirectories on Unix or Linux?

Answer 1: This will show the most recent 10 files in current directory and below.

It supports filenames with spaces. And can be slow with lots of files

http://stackoverflow.com/a/7448828

```
sudo find .  -type f -exec stat --format '%Y :%y %n' "{}" \; | sort -nr | cut -d: -f2- | head
```

```
2015-08-03 13:59:49.000000000 -0700 files/CV_Smith_1July2015.pdf2014-12-05 09:46:33.000000000 -0
```

Answer 2: This will show all files modified in last day, in current directory and below

```
find . -mtime -1 -ls
```

This version will just print the filenames, without the file sizes or times.

```
find . -mtime -1 -print
```

Thin Client technology - Linux Terminal Server Project HOWTO

The text below was copied with permission from post by Harry Mangalam, UCI, to UC-CSC Mailing List

I typed up the notes I took as I was evaluating the LTSP for Lab and Office deloyment, added a few introductory paras and got this:

http://moo.nac.uci.edu/~hjm/LTSP_HOWTO.html

Feedback, criticism, corrections, and suggestions how to improve it welcome. If you have additional links that should be added, let me know.

Those at UCI are welcome to visit for a demo until the hardware gets redeployed. I've tested with 64bit PCs netbooting 64 & 32bit OSs, and an old 32bit PIII laptop netbooting the 32bit OS.

From the introduction:

-————-

There are many technologies tempting your wallet these days with promises of secure, easy, efficient, low-cost, scalable desktop computing. Many of these are based on virtualized Windows, with proprietary technology at each layer of the whole solution. (Tom Holub and crew evaluated this Windows-based onion/parfait at Berkeley in 2008-2009.)

I say … meh.

Linux and the Linux Terminal Server Project (LTSP) have provided this

kind of technology for at least a decade for free. It is being used in large rollouts in the 3rd world where the technology cost issue is most important and in the 1st world where the human costs are the main issue. Before you sink $thousand$ trying to stack proprietary technology upon proprietary technology, why not see what some free software can provide? If you have the (minimal) hardware, you can have an LTSP system up and running in about 2 hours. No licenses, no signatures, no faxes, no crippleware, no POs, no RFPs, no bids, no lawyers. And surprisingly few tears.

Table of Contents1. Introduction2. Assumptions3. Pre-Requisites3.1. Prep work3.2. Hardware3.3. S

combining unix stderr output with stdout

The problem is that when I run script x and redirect it to a file, all of the error messages come to the screen when I want them to also go to that file.

With bash shell, use:

program > outputfile 2>&1

My colleague found this for me in the book Unix Power Tools which is available online at UCLA here. http://proquest.safaribooksonline.com/0596003307

How do I keep color output when paginating shell output through less?

Let’s say that you like the color output that comes out of ls (different colors for different file types). But when you have lots of files and want to paginate it through less, you lose the color output. So how do we keep the colors?

You need to give the -r flag to less to tell it to pass through the color escape sequences:

```
ls | less -r
```

That might not work because on some systems ls is set as an alias to:

```
ls –color=auto
```

(you can find out by typing “alias ls”)

That makes it show color only for interactive terminals (sort of…) but piping into less isn’t an interactive terminal.

So you can do this:

```
ls –color | less -r
```

or

```
ls –color=always | less -r
```

That’s probably too much trouble, so you can set up an alias by

editing your ~/.bash_profile and adding this: (if you’re using bash as your shell)

```
alias lsl=‘ls –color=always | less -r’
```

You can change lsl to whatever name you like.

Or if you want a long directory listing, you can do something like this:

```
alias lsl=‘ls -alg –color=always | less -r’
```

Unix Cheat Sheets and Tricks

While Unix is an extremely flexible, powerful, and stable operating system, mastering it can require apprenticing yourself to an expert and monitoring every keystroke. This article will hopefully grow as others add their favorite Unix tricks.

The best trick is to learn to read *man pages* and understand them. If you can do that, you'll have Unix at your fingertips.

- 'The Art of Unix Programming' by Eric Raymond – *full-text available online*
- Learn 10 good UNIX usage habits – IBM
- Advanced Shell Topics
- Slicing and Dicing on the Command Line – *Linux Magazine article on a host of methods for reformatting plain text — including the text used by graphical applications like spreadsheets and email programs.*

- **Cheat Sheets**
  - http://www.sloppycode.net/nix/

Searches (and Replace)

- Search and replace text in a list of files: perl -pi.bak -e 's/OLDSTRING/NEWSTRING/g' FILELIST
- search for tablename references in db program:
  - egrep -rl -e '(select|insert|update)(.*)tablename' *

- Extract username and realnames from passwd file: **cut -d: -f1,5 /etc/passwd | sort**

- Extract unique IP addresses (first field) from web server logs. In this case, any looking for

winnt to try and track nimda worm.

- tail -10000 access_log | grep winnt | cut -d" " -f1 | sort -u

Finding Files

- find files owned by group xxxx: **find . -group xxxx -print**
- find files owned by user xxxx: **find . -user xxxx -print**
- list most recent logins (and FTP): **last**
- list most recent logins (and FTP) by username: **last username**
- to change date/time of file:

  /usr/bin/touch -am -t 195401010000 filename _makes it Jan 01 1954 _
- show environment of all processes: **ps -ae**
- list table of contents of tarfile **tar tvf tarfilename**
- To change permissions on directories, not files, recursively. Here's two examples:
  - **find . -type d -exec chmod 775 {} \;**
  - **find . -type d -exec chmod g+s {} \;**
- To change permissions on all htm files recursively:

  chmod -R 664 *htm
- To find and print out all files owned by user:
  - go to /etc/passwd to find user ID number
  - go to the directory where you want to start your search
  - type: **find . -user 30152 -print**
- To find and replace owner: **find . -user 30152 -exec chown newuser {} \;**
- Find all php files in this directory or below
  - Linux find . -name *php
  - Solaris find . -name *php
- Find all php files in this directory or below and display first few lines of each
  - find . -name *php -exec head {} \;
- Remove all files that end with xxx: find . -name *xxx -print -exec rm {} \;
- This one searches all php, pl and pm files for the term board_extras

  find . \(-name **php -o -name* \pl -o -name *pm \) -exec grep board_extras {} \; -print | less
- Find and chgrp to classweb for all files with other group ID

  find . -not -group classweb -exec chgrp classweb {}

  \; | less

- Find all files with rw-r—r—: find . -perm 644 -ls | more
- chmod on them with ok? y/n: find . -perm 644 -ok chmod 664 {} \;
- chmod on all of them: find . -perm 644 -exec chmod 664 {} \; ran as root. Important to have space after {}
- Find all files named ta.inc and run search and replace to change " <?" to “<?”
  - `find . -name ta.inc -exec perl -pi.bak -e 's/^ <\?/<?/' {} \;`
- Find all files named ta.inc and print first line only (to check results of above script)
  - find . -name ta.inc -exec head -1 {} \;
- Find any core files left on system{ find . -name core

Custom Printing

- Print two pages to sheet, nicely formatted, with line numbers:
  - a2ps —line-numbers=1 filename | lpr

Comparing Files

- Side by side diff is an excellent way to visually compare files: sdiff file1 file2 | more
- Side by Side diff only showing lines on left if they’re the same, and 250 cols instead of 130
  - sdiff -lw250 wwwboard.pm wwwboardjon.pm
- vimdiff – edit two or three versions of a file with Vim and show differences
- diff3 – Merging and More with diff3

This article was originally posted on the UCLA Programmers Wiki.

vi editing notes

vi is a Unix text editor that is almost always available, on any Unix or Linux system. Knowing how to edit in it means you'll always have an editor available, for editing config files or whatever.

To convert a unix text file to a windows/dos text file

:%s/$/\^M/

% – the whole file (same as 1,$)s – substitute/ – substitution delimeter$ – end of line\ – (1 backslash) escape next character^M – carriage return (enter ctrl-v ctrl-m)

Uppercase a file

%s/.*/\U&/

from: http://sites.netscape.net/ianjdonaldson/

Replace tabs with a comma in vi

:1,$s/\([TAB]\)/,/g

Replace tabs with two spaces. Do it with Perl:

perl -pi.anytabs -e 's/\t/ /g' isisboth.inc

deletes every line that contains "string"

:g/string/d

deletes every line that does not contain "string"

:v/string/d

or

:g!/string/d

delete every empty line

:g/^$/d

remove trailing blanks

:%s/ *$//g

Replace the following lines

create user user103 identitified by user103

create user user104 identitified by user104

create user user105 identitified by user105

with

grant connect to user103

grant connect to user104

grant connect to user105

:%s/^.**(user.**$\)/grant connect to \1/g

copy lines 1 through 5 after line 35

:1,5t35

copy lines 1 through 5 at the end of the file

:1,5t$

remove null lines in the file

:g/^$/d

Replace commas with Carriage Returns

:1,$s/,/CTRL-V then press Return and ^M will pop up, then /g

10/23/2001

Trying to do regex but even simple ones won't work

:%s#\(dec.**)#\1\1# Substitute pattern match failed**

:%s#\(dec.\)#\1\1#

Anything with . in it fails.

Found that we were pointing at the /usr/ucb version of vedit. Using /usr/bin/vi it works. Changed the alias to point to right one. This is a common problem where the man entry talks about the version you aren't using.

vi – :set ignorecase for case-insensitive searching

:set noignorecase

[/literal]

To do a quick indent of several lines -

Go to the first line that you want to indent and type in the number of lines after that that you want to shift. Then push SHIFT + > (the greater than sign) to move your lines to the right and click SHIFT + < to move your lines to the left (you might have to push the carrot button twice). To change the

indention size, you have to set the width in your vi config file. Go to your home directory and add the following line to the .exrc file: “sw=2”.

This article was originally posted on the UCLA Programmers Wiki.

Linux scp command examples

I’ve used scp for years, but didn’t know many of these options. The compress-before-transferring and recursively copying a subdirectory structure, look particularly useful.

- 12 scp command examples to transfer files on Linux

diff and patch

- The Ten Minute Guide to diff and patch
- Using diff and patch

UNIX Tutorials

Learn UNIX in 10 minutes

UNIX Tutorial for Beginners

How To Look Like A UNIX Guru

Best Practices

- Learn 10 Good UNIX Usage Habits
- Learn 10 More Good UNIX Usage Habits
- Three Books Every System Administrator Should Read – Matt Frye Linux Magazine -July 1st, 2009
- Unix commands and tools you just can’t live without – Sandra Henry-Stocker – March 31, 2013

Linux tee command

The Linux “tee” command is extremely useful in the right situation. This article gives some good examples:

- http://linuxaria.com/pills/linux-terminal-the-tee-command

Manipulating data on Linux

This is an introduction to those using Linux for data analysis for the first time.

It came from a post by UCI colleague Harry Mangalam to UC-CSC Mailing List

“As an aid to helping students accomplish the basics on one of our clusters, I’ve written up a doc that describes some of the tools and utilities (and gotchas) that they might encounter. You’re welcome to use, re-use, or modify it as per the Creative Commons Documentation license. Suggestions, additions, edits, are welcome.”

A work in progress, it’s available here:
Manipulating Data on Linux

Advantages of less over more (UNIX)

This article is intended to provided reasons to use the UNIX program *less* over its predicessor *more*.

What is *less*?

If you use the *whatis* program for *less* on a UNIX terminal you get “opposite of more”. What does this mean? *more* allows a user to incrementally display text from the beginning scanning forward. *less* allows a user to view text backwards (in addtion to forwards).

Why use *less*?

less, in addition to forwards-backwards scrolling, also supports sideways scrolling. This gets rid of the wordwraping that *more* uses.

The controls are simpler. The arrow keys control movement. In addition the old controls work as well.

Another advantage of *less* is that it can start up before reading the entire file (which *more* has to wait for).

Useful features of *less* /keyword performs a search for keyword ?keyword performs a search for keyword backwards n repeats last search N repeats last search backwards g seek to beginning of file or line specified (ie 20g)^1^ G seek to end of file or line specified (ie 20G)^1^ m marks position (ie mz) ’ returns to position (ie ’z) s save the input if less input is a pipe (s file.txt) p or % jump to point in file. value must be between 0 and 100 (ie 20%)

[1] Warning: may be slow if number specified is large

Useful commandline options -N prefix line numbers to every line -jn jump to line n -xn[,m]* set tab location (ie less -x5,10,12 will align tabs to 5th line, 10th line, 12th line, 14th, 16th, 18th lines... because the difference of the last two is 2)

More help can be found using the following lines:

```
man less
less --help
```

Case-insensitive searching in vi

By default, all searches in vi are case-sensitive. To do a case-insensitive search, go into command mode (press Escape), and type `:set ignorecase`. You can also type `:set ic` as an abbreviation.

To change back to case-sensitive mode, type `:set noignorecase` or `:set noic` in command mode.

If you tend to use this feature a lot, consider putting `set ignorecase` into your vi configuration file. See the article on customizing vi.

Customizing vi

The Unix editor vi lets you customize its behavior in a numer of ways. There are two ways to customize vi. The first way is to enter command mode by hitting Escape, then type a colon and the command as it is shown.

The second way is to create a file called .exrc in your home directory. Each command goes on its own line, and the commands are parsed as though you typed them directly in vi. There are a couple rules your .exrc needs to follow:

- No blank lines are allowed.
- Directives must be typed on a single line.
- Lines beginning with double quotes (") are marked as comments and are ignored.

Setting options

vi has some options which take values, and other options which are either on or off. To set an option with some value, type `set [option]=[value]`. To turn a boolean option on, type `set [option]`, and to turn it off type `set no[option]`. Here are a few examples which will demonstrate the syntax as well as list some useful options.

- `set tabstops=4` – sets tabstop width to 4 spaces
- `set autoindent` – turns on auto-indent
- `set ignorecase` – ignores case for searching
- `set showmatch` – highlights matching parentheses and brackets when you type them
- `set nowrapscan` – searches will not wrap around to the beginning when they reach the end

Most options have abbreviations. You can find a full list of options, their abbreviations, and what they do at: http://ex-vi.sourceforge.net/ex.html#11

Abbreviations

By setting an abbreviation, vi will replace any occurance of the abbreviation you type with the full phrase. To set an abbreviation, type `abbr [abbreviation] [full phrase]`. For example: `abbr _ssc Social Sciences Computing` will take effect whenever you type "_ssc". You don't need to type any special commands, it will be replaced as you type.

Mappings

Mappings are just like abbreviations except they work in command mode rather than typing mode. You can set a map to perform a common task, like common replacements or insertions. You can put as many commands as you like into the mapping. The syntax for mappings is: `map [abbreviation] [command list]`

An example of a useful mapping would be one that spellchecks a word:
`map teh :%s/teh/the/g`

This executes a document-wide replacement command (teh → the) whenever you type "teh" in command mode. Note that you can use colon-prefixed commands here too. Also note that a better way to solve the above spellchecker would be to say: `abbr teh the` and vi would spellcheck as you type. ;)

A sample .exrc file which demonstrates a lot of features can be found at:
http://grox.net/doc/unix/exrc.php

PHP Client can break `less`

When you do php through command line (i.e. `php somescript.php`) and you pipe it to less (i.e. `php somescript.php | less`), this has the possibility of breaking the functionality of /usr/bin/less. To exit you need to type ‘q’ then hit enter.

How to synchronize files and directories from one location to another on Unix

Although tar can be used to synchronize/copy files/directories from one location to another (i.e. “tar -cf – -C srcdir . | tar -xpf – -C destdir”), it is really an archive tool and does not do the job as well as rsync. rsync does not have to go through the archive stage and can be used to synchronize one filesystem (or part of it) from one location (local or remote) to another (local or remote). rsync can do full or incremental synchronization. Although rsync is available on Unix, Mac and Windows, just the Unix usage will be described below.

To do a full sync of one location to another:
rsync -Havx srcdir/ destdir

To do an incremental sync after a full sync above:
rsync -Havx srcdir/ destdir (same command)

Note that for incremental sync, data added at the source (after the initial full sync) will be copied to the destination, but data deleted at the source will not be deleted at the destination. To force an incremental sync to delete data in the destination that is no longer in the source (this is like a real sync that is sometimes dangerous), do:
rsync -Havx —delete srcdir/ destdir

If you’re using a filesystem with a snapshot feature, include the option:
—exclude=.snapshot/

avoiding increasing indents while pasting into vi

When pasting text into a vi editing screen, sometimes you'll get each line increasingly indented like this.

```
aaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbb  ccccccccccccccccc
```

Use this command to change the "paste" setting. Then change it back when you're done.

`:set paste`

`:set nopaste`

SAS on 64b Ubuntu Linux

This was contributed by a colleague from UCI, Harry Mangalam.

Having spent 3 days debugging this, I thought I might make it easier for others who might run into it.

SAS 9.2 uses Java for at least some of its plotting routines (minimally the ‘ods graphics’).

The 64b version of SAS still uses the 32b version of Java and officially only supports SUSE and RHEL as a platform. For a variety of reasons, I run it on Ubuntu Intrepid, which meant that I got a flurry of "we do not support that platform’ replies from SAS technical support when I tried to figure out why it was failing with “ERROR: Cannot load Java Runtime Engine”

The short version is that in order to support the 32b version of Java, the 32b compatibility libs are required. You can install them on a Ubuntu platform with:

sudo apt-get install apt-get install ia32-libs ia32-sun-java5-bin\
sun-java5-jre libc6-i386 lib32gcc1 lib32z1 lib32stdc++6 lib32asound2\
lib32ncurses5

(if you’re going to use a direct-from-Sun JRE, omit the packages with ‘java’ in the names.).

I installed Sun’s latest 1.5 JRE (jre1.5.0_21) in the same SAS root as their supplied JRE (jre1.5.0_12). It seemed to work with SAS’s JRE, but it threw a few “Locking assertion failure” errors. Using Sun’s JRE, it ran without errors.

It also needs the environment vars set to tell SAS where to find things:

```
# convenience shortcutexport SASPATH=/where/you/rooted/SAS-x86_64/9.2# following is required to
```

SAS tech support spent 3 days insisting that it was the wrong Java sub-version number that was the problem.

—

Harry Mangalam – Research Computing, NACS, UC Irvine

Ruby one-liner to count logins in Linux last command

count logins in last command

If you want to take the output of the Linux `last` command, and get a count of how many times each user connected to your server, this script will do.

sample input

```
tommytrojan      sshd         abc.def Fri May 15 13:30 - 13:30  (00:00)joebruin      sshd
```

sample output

```
Total entries: 4joebruin = 2janebruin = 1tommytrojan = 1
```

script and explanation

```
last | ruby -ane 'BEGIN{rows=0; a={}}; next if $F[0].nil?; a[$F[0]]||=0;a[$F[0]]+= 1; rows += 1;
END{puts "Total entries: #{rows}" ; a.sort_by{|k,v|v}.reverse.each{|k,v| puts "#{k} = #{v}"} }'
```

- ruby -e executes the following script; -n loops; and -a splits input and outputs into $F array
- BEGIN block is necessary to initialize row count and a hash, where we'll store counts
- next if $F0.nil?; is necessary, because at least one of our input lines had no user field.
- $F0 gives us first field in input line.
- a[$F0]||=0; initializes hash location to 0 if it hasn't been used yet
- a[$F0]+= 1; increments user access count in hash
- END block lets us print our report.
- a.sort_by{|k,v|v}does a sort by the counts, not by the user names. a.sort would have been enough if we just wanted it sorted by the key, ie: usernames.

- reverse.each{|k,v| puts “#{k} = #{v}”} reverses the sort to get descending order, and outputs the key (k) and value(v), e.g. username = access count.

useful links

- http://benoithamelin.tumblr.com/ruby1line
- http://www.tutorialspoint.com/ruby/ruby_predefined_variables.htm

NOTE: My colleague did the same thing in this shell script. And his script includes the line with the missing user field.

```
last | cut -d" " -f1 | sort | uniq -c | awk '{print $1" " $2}' | sort -nr
```

Linux performance analysis in 60 seconds

Here’s an intro to understanding what’s going on with a Linux server, quickly.

http://techblog.netflix.com/2015/11/linux-performance-analysis-in-60s.html

If anyone has others, please post them here.

Reasons to Use Vim over Vi

This page Difference Between VIM and vi goes over some of the advantages of vim over vi.
(Note: the page disappeared, so now we're pointing at the Internet Archive's March 2011 cached version.)

To list some useful features:
Multi-level History

- Undo – *n* u (*n* is the number of levels)
- Redo – *n* ctrl-r

Windowing

- horiz split – :new *file*
- vert split – :vs *file*
- next window – ctrl-w ctrl-w
- prev window – ctrl-w ctrl-p

Visual mode

- start – v
- finish and…
- move – h,j,k,l
- change – c
- delete – d
- yank – y
- alter case – ~,U,u

Command Completion – *tab*

Insert Completion

- match dictionary – ctrl-x ctrl-k
- match from includes – ctrl-x ctrl-i
- match line – ctrl-x ctrl-l

- current file
- next match – ctrl-n
- prev match – ctrl-p

Additional info: http://csswizardry.com/2014/06/vim-for-people-who-think-things-like-vim-are-weird-and-hard/

Unix grep, find and maxdepth

I'm writing this so I can find it next time I want to do this, and so my student programmers can more easily compare config settings across test sites.

Problem: Find a particular line quickly in a file in 7 different directories. In this case, it's a Moodle config setting in 7 different installs, each with thousands of files.

Solution: With the help of this site, http://helpdesk.ua.edu/unix/tipsheet/tipv1n10.html I found a quick way to do it without traversing the thousands of files in each directory. The cheat is that I knew the config file in a Moodle install is always in the same spot. But I could leave maxdepth out.

The key here is that while I know

```
grep -r search_string *
```

would work, it would take forever to go through all of those files, some of them quite large. I didn't know how to get the recursive grep to only look at certain files. Turns out that is what backticks are for.

```
grep quiet `sudo find .  -maxdepth 3 -type f -name config.php`./stage/moodle/config.php:
```

Quick Explanation:

1. grep quiet – will search for the text "quiet" in the list of files produced by the command in backticks.
2. sudo – to avoid being told I don't have permission to see certain files.
3. maxdepth – to only go 3 directories deep instead of recursing through everything. (I knew where the config.php file would be.)
4. -type f -name config.php – only look at files named config.php. Ignore all others.

Useful Commands for Vi

The page is meant to document some of the more useful, but obscure commands in vi. For commands exclusive to vim see Reasons To Use Vim Over Vi. Please feel free to add commands you would like to see on this page.

% move to matching (, {, or [*n* G goto line *n* f *char* move to position of the next *char* t *char* move to position before next *char* J join the next line to the current one m *reg* mark position in *reg* which can be a-z ` *reg* move to position in *reg* which can be a-z

Notes:
f and t are useful for changing quotes. say you are at the first position on the line below

echo “old_text” . $var . “preserve”;
to change this you simply to say “new_value” in place of “old_text” you can press

f"lct"new_value
m and ` are useful for marking locations to remember and moving large chunks of code.

using find and chmod together in Unix

In Unix, if you need to clear up a set of subdirectories to see what permissions or ownership of everything under a given directory, these commands work for me on Solaris. There are differences because you don’t want to give execute permissions to files. And these commands assume that none of the files need to be executable.

Directories

This command finds and lists owner, group and permisions for all directories beneath the current directory.

```
find . -type d -exec ls -algd {} \;
```

This command changes the permissions of all of them. Obviously, only use 775 if those are the permissions you want to give.

```
find . -type d -exec chmod 775 {} \;
```

Files

This command finds and lists owner, group and permisions for all files beneath the current directory.

```
find . -type f -exec ls -alg {} \;
```

This command changes the permissions of all of them. Obviously, only use 664 if those are the permissions you want to give.

```
find . -type f -exec chmod 664 {} \;
```

Alternative Syntax

If you only need to change permissions within a single directory and not all subdirectories these commands will work as well:

This command will change the permissions of all files in the current directory to 664
`chmod 664 *`

Immediately followed by this command:
`chmod 775 */`

which will change the permissions of all directories to 775

finding words in binary files

In Unix the strings -a command will output all the text strings in a binary file.

Solaris man page for strings says: “The strings utility looks for ASCII strings in a binary file. A string is any sequence of 4 or more printing characters ending with a newline or a null character.”

This was extremely helpful just now while trying to debug a program where we didn’t have the source code. In this case it let us find all references to spss.

```
strings -a subsda3 |  grep -i spss
```

Apple OS/X also seems to have the “string” command.

grep tricks

I’ve been using *grep* (Global Regular Expression Print) to search files on Unix for years and never knew you could have it appear in color. And here are some other interesting articles about *grep*.

- http://unstableme.blogspot.com/2009/03/highlight-match-with-color-in-grep.html
- http://opensourcehacker.com/2012/05/29/power-searching-using-unix-grep/
- http://eriwen.com/tools/grep-is-a-beautiful-tool/
- http://www.itworld.com/operating-systems/364065/unix-groping-through-big-data-grep

http://regexr.com/ is an online tool to learn, build, & test Regular Expressions.

Location of the sudo log file

The log file contains the commands issued along with the issuers’ user names.

Distribution Location Centos /var/log/secure