UNM ACM: Understanding Linux Session **October 7, 2003**

Linux Tips You'll Use - Aaron Clauset

1.1 manifesto

this document is intended to be a set of linux (and more generally, computing) tips that you can actually use to get more/better work done. it is a compendium of bits and pieces which you're likely to eventually pick up on your own. but why wait? keep this handy to refer to later or at least read it and know what's out there. unix is an incredibly powerful OS and the more you know about what it can do for you, the more you'll be able to do with it.

1.2 playing nice with other users

the linux machines in the department are a *shared resource*, and a much larger population of people use them than simply those who sit down in front of the machine; the wonderful thing about computer science is that people can, and often do, work remotely. by *remotely*, this can mean either remotely in time (running a job in the background or scheduling a job) or space (logging in from elsewhere). generally, the two shared resources are the CPU (time) and the disk/RAM (space); here are three ways to be kind to your fellow users:

1. do not reboot any shared machine, as there are likely to be other users concurrently logged in or who are running processes which will be terminated upon reboot
2. do not log out using Crtl-Alt-Delete; this may cause some processes to be *zombied*, which will consume 100% of the CPU, making a machine significantly less useable for others. instead, logout using whatever mechanism is provided by your GUI environment
3. `renice` your long running processes so that others can use the machine for their work
4. if you use a GUI environment (like kde or gnome), clear the caches periodically as they can slow a machine for other users
5. logout when you are done using a machine; otherwise

1.3 how to use cssupport effectively

1. **let them know** - the `cssupport@cs.unm.edu` address is great for alerting them about things that go wrong with the compute resources. they *like* when people send them

emails about it. it's also a great way to get help with stuff, but it's in your best interest to observe the following rules of thumb:

2. **be nice to them** - cssupport spend most of their time fixing problems and dealing with things that don't work as expected, which you can imagine gets tiring after a while. it's unfortunately a thankless job with impossible expectations to meet. however, they are also the people who help you out in the few cases where you have a deadline or something and really really need something done. makes friends with them!

3. **if it's urgent, go in person** - generally, if you need something addressed immediately, you'll get a faster response if you go ask for help in person.

4. **but try to figure it out on your own first** - a lot of standard questions people have about linux/unix are answered online, and while it may take a bit more time to track the answer down yourself, you'll be a lot better off knowing yourself (because it means you won't have to waste time asking someone else in the future).

1.4 file, process and code management

1. **file management** - creating a logical (to you) directory substructure in your home directory can be a great boon to your productivity. develop a system for storing/categorizing files and stick with it. don't forget to delete (`rm`) unneeded files and to periodically compress, archive and backup important data.

2. **process management: renicing** - if you plan on using a shared machine (such as the ones in fec309, the computer lab) to run your own programs, you **must** be *nice* to other users, as they have the same right to use the machine as you. running several CPU-intensive processes for more than a few minutes with the default *nice* level (0) will bring a machine to a grinding halt and make it largely unusable to others. see below for an explanation of how to use the `renice` command.

3. **code management** - concurrent version system (cvs) is a code management tool that provides excellent ways to share, archive and log changes to code, cvs is extremely handy. for more information, see `http://www.cvshome.org/`. lacking that, it's a good idea to make frequent backups of your code and archive them somewhere - although infrequent, having to revert to a previous code version can be really a pain if you haven't archived in a systematic way - develop your own system and stick with it.

1.5 making unix work for you

*nix is powerful and flexible if you know how to use the command line. gui environments like kde, gnome, etc. can be useful, but the more functionality they give you, the more

space (memory/disk) and time (cpu) they require. also, a large number of tasks can be accomplished more quickly from the command line, than with a mouse. the more comfortable you are with the command line, the more productive you will be in a *nix environment.

1. your terminal shell (`bash`, `csh` or other) - choose a shell type (or go with the default). the only time it matters which you use if you edit your `.bashrc` or `.cshrc` files, as the syntax differs somewhat. for help in learning how to customize your terminal experience (i.e. aliases, color, custom prompts, setting environment variables, etc.) google `.bashrc` or `.cshrc` and look at the examples it turns up.

2. scripting - perl has become the standard scripting language in *nix environments because of its flexibility and power. there are plenty of tutorials online and many good books about it; to adapt a popular saying: give a man a script and he'll be productive for the rest of the project, but teach a man to script and he will be productive for life.

3. 26 useful utilities - with only a handful of command line commands, you can do quite a lot. the following are the basics, along with a few hints about using them effectively:

   (a) `^C` - (control-c) the standard *break* command for a terminal. use this to abort a process run from the command line that has not been *backgrounded*

   (b) `./path` - runs an *executable* file denoted by `path`, regardless of whether it is in your `PATH`

   (c) `command &` - runs a process in the *background* so that the terminal issuing the command can still be used

   (d) `cp sourcePath destinationPath` - standard copy (duplication) from source to destination. you can use *wildcards* in the source.

   (e) `df` - disk space free. displays the usage of disk space on the various drives mounted. use `df -h` for values in human-readable format (i.e. K, M, G).

   (f) `du` - disk usage. recursively traverses the directory structure from the current directory, showing the total disk space used for each subdirectory and returning a grand total at the end. use `du -h` for values in human-readable format (i.e. K, M, G).

   (g) `grep pattern filename` - a way of printing lines in `filename` which match the `pattern`. commonly used in conjunction with `cat`, i.e. `cat .plan | grep 505` (to grab my office phone from my `.plan` file) or `ps -ef | grep mozilla-bin` (to display all the `mozilla-bin` processes (with username and pid) on the local machine)

   (h) `kill -signal pid` - the single most useful utility available. `kill` sends signals to processes that you own. the most common signal you'll send is `-9`, which is the terminate signal. the `pid` is the process identification number, which you can get from either `top` or `ps`

(i) `ls option` - displays the contents of the current directory. use the `l-l` option to also list file sizes, permissions and modification dates. use the `-a` option to list all files (including hidden files).

(j) `lpr printer filename` - print the file `filename` to printer `printer`. for instance `lpr -Pfec309 pentagonSecrets.txt`. note that you cannot use `lpr` to print `.pdf` files - you must use `acroread` or `xpdf` first and print from inside the application. further `mpage` can be used to print multiple pages on a single sheet of paper. all printers, by default, now print *duplex* (double sided) and omit the header page. if you want to print *simplex*, append `s` to the printer name, i.e. `lpr -Pfec309s PnotequalNP_proof.ps`

(k) `lpq printer` - check the queue of print-jobs on `printer`, i.e. `lpq -Pfec309`

(l) `man command` - man pages are unix help files, where `command` is the name of the unix command you're curious about. `man` can also be used to look up common C/C++ libraries and functions. they're not always that helpful, and you're often better off searching Google, but if you just need a reminder or are looking for a particular parameter, `man` is the way to go

(m) `more`, `less`, `cat filename` - these display the contents of a file (usually text files). `less` is usually favored over `more`, and both display a screenfull at a time. `cat` displays all contents and is useful for redirecting its output to other using functions use the *pipe* `|` command

(n) `mv sourcePath destinationPath` - standard move (duplicate and delete original) from source to destination. you can use *wildcards* in the source.

(o) `nohup command` - normally, when you close a terminal session, all processes which were initiated from that terminal will *hangup* (terminate). this includes *backgrounded* processes. `nohup` tells the OS to not do this to the process created by `command`. commonly used to start simulations/programs that will run to completion without interaction: `nohup ./ruleTheWorld &`

(p) `passwd` - change your password

(q) `ps` - displays the processes which you own running on the local machine

(r) `pwd` - displays the current directory

(s) `renice value pid` - processes with the same *nice* value are given equal priority to the CPU scheduler, while values closer to $-\infty$ are given proportionately higher priority; if you expect your process to run for a long time and consume large portions of the CPU's attention, please renice it to at least a `value` of `10` (values are integers from $[-20 \ldots 20]$). note that you may only `renice` a process to a higher `value`

(t) `rm` - delete a file. use `rm -rf directory/` to recursively delete a directory and its contents.

(u) `scp source destination` - secure copy of files between machines. either the `source` or `destination` paths will be of the form `username@host.domain:path/`. note that you can use *wildcards* in the `source`; if you are retrieving files from a remote machine, then you need to enclose the entire `source` in single quotes

(v) `ssh username@host.domain` - secure shell remote login from a terminal to a remote machine, i.e. `ssh aaron@santafe.cs.unm.edu`

(w) `tail`, `head filename` - these respectively display the end and the beginning of a file `filename`. particularly useful is `tail -f`, which *watches* a file and displays lines as they are added

(x) `tar -cv inputPath/ -f outFilename` - standard combination (and optionally compression) utility. the `inputPath/` denotes the directory or file you wish to tar up, while `outFilename` is what you want the final thing to be called. including the `-z` option uses `gzip` to compress the *tarball* before giving it the final `outFilename`. convention is to give uncompressed tar files the extension `.tar` and compressed tar files the `.tar.gz` or `.tgz` extensions

(y) `top` - essential to see what processes are running a machine, how much CPU and RAM they're taking, who owns the most resource hungry processes, etc. depending on what parameters you pass it, you can customize the information it displays

(z) `wc option filename` - count things in a file: words(`-w`), line(`-w`), characters(`-m`), bytes(`-c`), newlines(`-l`) or the length of the longest line(`-L`)

1.6 available compute resources

there are a large number of machines available for use both for general computing (i.e. for course work) and a smaller number available for shared research uses. generally speaking, the lab machines are both the ones you'll want to use for projects and also for your checking your email. be careful about where you login to check your mail, because older machines occasionally get cycled into student offices and become personal workstations.

1. **general usage/login**: (fec309 machines) `anderson`, `columbia`, `husband`, `mccool`, `clark`, `chawla`, `brown`, `ramon`, `alamogordo`, `carlsbad`, `farmington`, `riorancho`, `lascruces`, `albuquerque`, `clovis`, `roswell`, `santafe`, `hobbs`, `hancock`, (`mountain` machines) `sangredecristo`, `sacramento`, `sanpedro`, `sandia`, `manzano`, `capitan`

2. **research only**: `camaross`, `shelby`, (the newgigs) `camden` (newgig-0), `lexington` (newgig-1), `ticonderoga` (newgig-2), `trenton` (newgig-4), `princeton` (newgig-5)