An Introduction to Mathematics Computer Facilities

SEPTEMBER, 2011

Contents

I. Overview of the Department's Computers

The main departmental server is *coxeter.math.toronto.edu* (usually simply called *coxeter*), a quad processor SunFire X4200, which is used for accessing email, TEXing and printing documents, reading email, etc. The department also has another quad processor SunFire X4200 with 16 GB of RAM, *sphere*, which is a compute server for running users' programs or various symbolic calculators. There are also incoming and outgoing mail servers, *mail* and *smtp*, (running encrypted IMAP and message submission, and a webmail service), a web server, *www*, running the SilverStripe CMS (content management system), and *blog* and *wiki* servers. The department also owns a number of micro-computers mostly running Linux and Microsoft Windows.

These computers are linked in a network which also includes a large number of personal computers in faculty members' offices. Also on the network are machines in the computer room, BA6200. There are two main public laser printers (both Xerox Phaser 4510 printers) on the system: `lw2` in the mailroom (BA6290A) and `lw3` in the computer room (BA6200). See Chapter II Section 12 of this document for more information about the local `Print` command for using the printers (you can just run `Print` on *coxeter* for a summary of the print options). Accounts for new users are made on the *coxeter* and on the compute server *sphere* and access is automatically given to the mail server (*mail*).

The email address `requests@math.toronto.edu` is for computer systems management questions. Use the address `consult@math.toronto.edu` for general computing help. For example, use `consult@math.toronto.edu` for questions about TEX, Maple, Matlab, Mathematica, Linux, Windows, or Macintoshes. Use `webmaster@math.toronto.edu` for questions about the webserver (such as creating your own webpage). The main departmental webpage is at `http://www.math.toronto.edu/`, and you can reach some computer help by choosing the "About" section near the top of that page, and then the "Computing" link on the left-hand side of the page. (`http://www.math.toronto.edu/cms/computing/` is the direct link.) Also, there is a very important computer systems blog for the department at `http://blog.math.toronto.edu/itstatus/`. The latest information about hardware and software changes (including downtimes) will normally be posted there.

The main compute server, *sphere*, has Mathematica (the latest version, 8.0.1), Maple, Matlab (the latest version 7.12.0.635 (R2011a)), and some other mathematical software. Type **bigjobs** on *coxeter* for information on running processes with appropriate settings so as not to unduly inconvenience other users and to plan for server outages. Use *coxeter* for word processing and other less computationally intensive tasks. It is the main departmental server supporting our TeX family users, news, and web access via **firefox** on X-window systems or **lynx** on ASCII terminals. *mail* is the incoming electronic mail server (which is only accessible via an encrypted IMAP connection to port 993 or via secure webmail), and *smtp* is the outgoing SMTP electronic mail server (which is only accessible from the departmental networks or if you use a secure authenticated TLS connection to the message submission port 587). The offices in Earth Sciences and the graduate facilities in 1 Spadina, Sidney Smith Hall 622, and the ninth and tenth floors of 215 Huron contain several machines for connecting to the departmental systems, as well as their own printers (`lwesps`, `lws`, `lwss`, `lwh`, and `lwh2` respectively).

You can arrange for links from the appropriate departmental pages to personal webpages or to pages that you create for courses. If you wish to have such webpages then please send email to `requests@math.toronto.edu`. Questions/comments concerning webpages should normally be sent to `webmaster@math.toronto.edu`.

The following is a brief list of some of the software running on departmental machines. The main servers have C/C++ and Fortran compilers. **mathematica**, **matlab** and **maple** are available on *sphere*. All the Linux machines have TeX and *Emacs*. Some public domain math programs are available on *sphere*, including **M2** (Macaulay2), **octave**, and **gp** (Pari/GP). On *coxeter* there is **xfig** and **tgif** which are drawing programs for X-windows that produce postscript or LaTeX output. For a more complete software list, type **unixmachines** on *coxeter*.

II. UNIX

1. Introduction

What follows is a brief general introduction to UNIX, which is the operating system running on *coxeter*, and on the Linux machines.

An operating system is the control program for the computer; it is in charge of allocating the machine's resources. In the UNIX operating system a user begins communication with the computer by "logging in", which consists of entering login name and password at a terminal. UNIX then starts up a program called a "shell" through which the user types in commands to be executed by the computer. When you enter a command at a terminal, the shell interprets the command and calls the program you want.

Most computer operations manipulate either the computer's main memory (RAM) or its storage area (hard disk). Roughly speaking, main memory is where you do your calculations and the hard disk is where you store any output that you wish to access at a later date. Data on the disk is created and manipulated in chunks known as files. A file is a collection of information such as the text of a report, C program code, or executable code created by a compiler. Each file is stored under a unique name assigned to it by its creator and is subsequently referred to by name. The UNIX system includes several hundred utility programs which perform functions commonly required by users. UNIX commands give you the ability to create, display, edit, print, copy, search, sort and delete files.

Warning: Generally speaking, in the UNIX operating system any command you issue which creates a file with a particular name will destroy a previously existing file by that name if there was one.

The UNIX file system provides a structure where files can be arranged into logical grouping of related files called directories. Directories can in turn have sub-directories, and so forth, in a tree-like organization. This structure helps users keep track of their work. Each user has one primary directory, called the home directory, and as many subdirectories as they decide to create.

2. Getting Started

To get started, you need to know two things:

- your login name, which is the unique name that identifies you to the system.
- your password, which protects your files from other users, much as a PIN number protects your bank accounts from unauthorized access.

These can be obtained from Marco. To log in, type your login name and password at a terminal. Your password will not be echoed on the screen, in order to protect you. If all is well you will be greeted by the "message of the day" (if this scrolls off your screen too quickly you may type **motd** in order to read it again) and then a prompt will appear, indicating that your shell is active and waiting for you to type a command. If anything goes wrong with the above, and you find you can't log in, see Marco.

Warning: *Do not allow other people to use your account. Protecting your password is important since anyone with access to it can destroy your files or impersonate you completely. It is also safer for everyone in the department if there is no unauthorized access to our system.*

If this is your very first login you will be immediately forced to change your password. Subsequently you may change your password again at any time by entering the command **passwd**. Pick one that is different from the one you were assigned by Marco, easy for you to remember, but not easy for others to guess. (For example, your first name or telephone number would be poor choices.) You will have to type it twice, to ensure that you typed what you intended. After this is completed, you will receive the prompt again. Note that password information is passed on from *coxeter* to the other departmental servers, so all passwords must be changed on *coxeter* and will propagate to the others shortly thereafter.

On successful login, you are given a shell, and your current directory is initially set to be your home directory. You are now ready to start typing commands. As you type a line, you can make some rudimentary corrections, like backspacing and retyping, but once you hit the ENTER (or return) key, the line is sent to the shell as a command. The shell then performs "command line substitutions" on what you typed (see below) and submits the result to the computer for execution.

To quit your terminal session, you type **exit** or **logout** or *Ctrl−***d** (that is, you hold the *Ctrl* key down and hit the *d* character). It is very important that you quit your session properly; otherwise the next person to sit at the terminal you were using will have access to your account and this person could, for example, remove all your files, which could prove very inconvenient.

3. Commands

A command in UNIX is executed by typing a command line beginning with the name of the command. The other words on the command line are called parameters (or arguments) and their interpretation will vary from one command to another. Most UNIX commands can be given parameters of the form "$-x$" (called options or flags) which slightly modify their behaviour. For example: **ls**   produces an alphabetically ordered list of the files in a directory;   **ls -s**   produces the directory listing with the number of kilobytes (a byte is the amount of memory occupied by a single character, and a kilobyte is approximately 1000 bytes) of storage listed for each file;   **ls -t**   produces the list ordered by date of last modification; etc. You can abort the execution of most UNIX commands (if you are fast enough) by typing $Ctrl-$**c**. You should receive the shell prompt back quickly after that.

4. File Names in UNIX

As mentioned above, in the UNIX operating system the files are organized into directories in a tree-like structure. At any time there is one directory known as your "current directory" or "working directory". Other directories are usually referred to by their position relative to your current directory.

   A file is identified in UNIX by giving its filename including either directly or indirectly the name of the directory it is in. The directory can be identified by its position relative to the current directory or by its position relative to the top of the tree. The top or *root* directory of the tree is written as "/". The notation $/a/b/c$ means the file named c in the subdirectory b of the subdirectory a of the root directory. A filename of this form, specifying the position of the file relative to the top of the tree, is called a *full pathname*. To be a valid filename, each stage of the description must be valid. For example, if the root directory does have a subdirectory named a, but a does not have a subdirectory named b then $/a/b/c$ is not a valid filename. Most UNIX commands are likely to respond to an invalid filename with "`No such file or directory`." A file can also be specified by a string which does not begin with /. In this case the name is interpreted as being relative to your current directory. For example, suppose your login name is *jones*. Then your home directory might be $/u/jones$. If your current directory is your home directory, $/u/jones$, then the filename $dd/junk$ is equivalent to $/u/jones/dd/junk$.

5. Some Basic UNIX Commands

The usual way to create a file is to invoke an editor, specifying the filename you want. The most popular editors used in our department are *vi* and *emacs*. Typing **vi** *myfile* or **emacs** *myfile* allows you to enter text, edit it, and save it in a file. Details on how to use *Emacs* are given in section IV of this document. Marco has "GNU Emacs Reference Card" and "Vi Quick Reference" documents if you would like to use *vi* and/or *emacs*.

When you save a file, it will be saved in your current directory. Like files, directories have unique names. To find out the name of your current directory, type **pwd** (print working directory). When you first start using the system, you have only one directory, your home directory, similar to */u/loginname*.

To find out what files are in your current directory, type **ls** (list). Some files do not show up when you type **ls**. These are special configuration files, whose names begin with ".". If you want to see these files, type **ls -a**. You should not attempt to change these files unless you are an experienced UNIX user — they have been set up by the system administrator to provide a suitable interface. If you change, or accidentally delete one of these files, your shell may start to behave in a very unfriendly manner, and could even become unusable.

Once a file is created and saved, you can see its contents (if it is a text file) by typing **cat** *filename* which places the contents of the file on the screen all at once. A better way to do the same thing is to type **more** *filename* which allows you to go through your file one screenful at a time by hitting the space bar when ready for the next page. You can type **q** when inside of **more** to "quit" if you don't wish to see the whole file. (Some people prefer to use the **less** command rather than the **more** command since it has more features.) You can get rid of a file by typing **rm** *filename*. Be careful with this one — UNIX will not let you change your mind, and your file will not be able to be recovered. You could easily lose several hours' worth of work. You can print a text file with the **Print** command. (See section 12 of this UNIX chapter).

You will probably want to organize your files into subdirectories fairly quickly. To make a new (empty) directory, type **mkdir** *newdirname*. An **ls** of your current directory will show this new directory in it, as if it were an ordinary file. However, you cannot **cat** or **vi** a directory. Instead, you can type **cd** *newdirname* (Change Directory). and you will be placed in that directory. You can now save files there. Conceptually, your directory structure now looks like:

$$/u/jones$$
$$|$$
$$newdirname$$

where */u/jones*, your home directory, is the parent of *newdirname* in the directory tree.

You can **cd** to a directory by giving its name. If you type **cd** without a name, you are placed back in your home directory. If you want to get to the immediate parent of your current directory, type **cd ..** . You can remove an empty directory by typing **rmdir** *dirname*.

To make a copy of *filename1* under the name *filename2*, type   **cp** *filename1 filename2*. Similarly you can type   **mv** *filename1 filename2*   to rename *filename1* as *filename2*. *Warning*: These commands will overwrite an existing file named *filename2* if there is one.

In summary we have:

cp *filename1 filename2*	copy filename1 to filename2
mv *filename1 filename2*	move (rename) filename1 to filename2
rm *filename*	remove (delete) filename
more *filename*	show contents of filename by screenfuls
less *filename*	show contents of filename by screenfuls
mkdir *newdirname*	create new directory
rmdir *dirname*	remove an empty directory
cd *dirname*	change directory
pwd *dirname*	print working directory
ls *dirname*	list directory contents

6. Command Line Substitution

The characters \, ~, $*$, ?, ′, ”, ‘, $>$, $<$, |, &, #, ;, %, ^, \$, [,], (, and) have special meanings to the shell and are called *metacharacters* (there are a few other characters which can be special to the shell; see **man tcsh** or **man ksh** for more information). If you submit a command line which contains metacharacters to the shell it will perform various substitutions before passing the command on to the UNIX kernel.

The shell metacharacters provide a facility for filename expansion based on a pattern matching system. This means, that if you want to apply a command to a list of similar filenames at once, you can specify this list using these special shell metacharacters. The actual list of files received by the kernel will be obtained as all files matching the pattern you specify. We will describe a few simple examples here.

The $*$ character will match any sequence of characters. So $z*$ matches all filenames beginning with z. Thus if you type   **rm** *, all the files in your current directory will be deleted, except those beginning with ".". (This a special exception built into the shell to protect configuration files.) Here are some more examples: $*.c$ matches all filenames that end with ".c"; $file*$ matches all filenames that begin with the letters f-i-l-e; $x*/*z*$ would match any file whose name contains a "z" and which is contained within a subdirectory whose name begins with "x". Much more is possible — for details, see a UNIX book.

One consequence of the use of metacharacters by the shell is that it is much harder to refer to the standard symbol represented by one of these characters while typing in the shell. For example, if you had a file named "x>y", you would not be able to refer to that file directly by typing its name. Consequently you should avoid choosing filenames containing any of the metacharacters or whitespace (spaces, tabs or newlines).

7. Input and Output Redirection and Pipes

Most UNIX commands take the characters typed at the keyboard as their input and place any output produced on the screen. To send the output of your command to a file instead of the screen, redirect the output using >. For example the command **date** prints the date on the screen, so **date** *>outputfile* creates a file called *outputfile* containing the date.
Warning: This will overwrite *outputfile*, if it already exists.

To append the output of your command to a file, redirect the output using ">>". Thus **cat** *myfile >>outputfile* will modify *outputfile* by adding the new output (in this case the contents of the file *myfile*) to the end of its original contents.

Similarly, if you want your command to take its input from a file instead of the keyboard, redirect the input using the metacharacter "<".

It is also possible to use the output of one command as input to the next command by means of a construction known as a *pipe*, invoked with the metacharacter "|". The syntax is *command1 | command2*, which means execute *command1* and then execute *command2* using the output from *command1* as input for *command2*. For example, to see the contents of a directory that contains so many files that the list produced by **ls** scrolls off the screen, you could type **ls | more** to see one screenful at a time.

8. Ownership and Permissions

UNIX allows for protection of user files via a system of permissions, based on the ownership of files.

If you create a file, you are its owner, and have the ability to set the permissions on it. In UNIX, the users are divided into groups. UNIX considers that there are three classes of people with respect to file permission: owner; group (i.e., same user group as owner); and all others. There are three kinds of activities that may be performed on a file: read; write; and execute. Read permission on a file is required to examine its contents; write permission is required to change its contents. The meaning of execute depends on the type of file. For a program file, execute means to launch the program; for a text file, execute is meaningless; for a directory, execute means to **cd** into it.

There are thus a total of $3 \times 3 = 9$ permissions that may be set:
$$\begin{Bmatrix} r & read \\ w & write \\ x & execute \end{Bmatrix} \text{ for each of } \begin{Bmatrix} u & owner \\ g & group \\ o & others \end{Bmatrix}.$$
You can see the permissions set on your files, and other information as well, by typing **ls -l** . Here is sample output from **ls -l** on *coxeter*:

```
-rwxr--r--  2 jones  math    816 Jul 05 08:34 file.c
drwxrwxrwx  2 jones  math    465 May 27 16:37 mydir
```

The first column tells you the file type and permissions:

```
-rwx r-- r--
```
└─ permission for others
└─ permission for group
└─ permission for owner
└─ type of file: - ordinary file; d directory

The second column tells you the number of references to the file, the third the owner's name, and the fourth the owner's group. The fifth column is the size of the file in bytes. The sixth, seventh and eighth columns give the time the file was last modified, and the last column gives the file name. The initial permissions for text files that you create is: `-rw-r--r--` meaning that anyone can read the file, but only you can change its contents.

To change the permissions on a file, use the chmod command:

chmod g+w *file.c*   (add write permissions for group.)

chmod o–r *file.c*   (remove read permissions for others.)

For more details about **chmod**, **ls**, and other UNIX utilities, type   **man** *command-name* to see the documentation.

9. The X Windows Interface

Everything we have discussed up until now has involved only interacting with your login shell via the command line. If you have connected via the internet this might be all that is available, but most machines, including the Linux machines within the department, provide an X-windowing interface which allows you to run mouse-driven programs and to run more than one shell simultaneously by offering you more than one window at a time. A simple windowing interface (such as mwm or twm) might put up a clock and a window; pressing the right mouse button in the background area outside this window brings up a menu which can be used to create more windows. There are also more complicated windowing interfaces available (e.g. gnome and kde) which place multiple icons on your screen for accessing various programs. Your windowing interface is determined by your "window manager", which is specified in your .xsession configuration file (see below). When using a windowing interface, remember that closing all of the windows is not always equivalent to logging out. If you do not quit the window manager, anyone who sits at that machine can use it to open up a window which is logged in to your account. From a simple windowing interface, quitting the window manager may be done from the menu created by pressing the mouse in background area. With one of the multiple-icon interfaces, one of the icons will provide the means to log out. After a successful logout, the terminal should reset itself within thirty seconds or so and redisplay its "Welcome" window. If this does not happen, please contact Marco.

In order to run X-Window applications your DISPLAY environment variable needs to be set to the display you are using. This is done automatically when you login directly to an X-Windowing system. To use X on a remote machine your DISPLAY variable must be propagated to that machine. See the discussion of **slogin -X** in Section 11.

The windows that you see on your screen have various abilities. You can resize, move, maximize, minimize, and iconify them. This can be done via a menu which appears when you move the mouse pointer to the top bar of the window, and click and hold the right mouse button. Windows may have scrollbars on the left, which allow a limited amount of scrolling through the lines you have typed.

Once you have more than one window on the screen, you must choose which window you want to type into by moving the pointer to the inside of the window, and you may have to click the left mouse button once. This is called giving the window "focus". The cursor will appear as a small solid box in the window that has the focus, and as an empty box in other windows.

10. Customization and Environment Variables

UNIX accounts can be configured by creating certain files whose names begin with a period and entering commands in them. New accounts on our system are provided with such configuration files when the accounts are opened. This section will discuss the different types of configuration files, and how to make minor customizations of your own if desired. If you do not create the configuration files described below, you will have the default settings.

Earlier we mentioned that in UNIX, communication with the computer is done through a program called a shell. There are in fact several shells used in the UNIX world although they have a lot of common features. Our Linux machines offer a choice of "C-shell", "Bourne shell", "bash", and "tcsh", while the IBM machines offers these plus the "Korn Shell", which is a modern variation of the "Bourne shell". Each user can configure his or her account to use any of the available shells, but we have configured new accounts to use "tcsh".

Shells are in fact programming languages and in them one can assign values to certain variables called "environment variables". Some programs will examine the value of some of these variables and adjust their behaviour accordingly. For example, the **more** program looks at the environment variable MORE. You can examine your environment variables with the command **printenv**. You can assign values to environment variables with the syntax **setenv** *variablename value* when using tcsh, or the syntax **export** *variablename=value* when using bash. So, for example using **setenv MORE -d** will have the **more** program prompt more verbosely than usual.

The most important environment variable is PATH, which is the list of directories that the shell searches for commands to execute. If the shell replies "Command not found" to some string you type, the precise meaning of this response is that there is no command by that name in any of the directories in your PATH, although it is conceivable that a command by that name exists somewhere on the hard disk.

The tcsh-shell configuration files are called ".login", ".cshrc", and ".logout". The commands in your .login file are executed automatically every time you log in. The commands in your .cshrc file are executed every time you open a new shell — for example by opening a new window under an X-windowing interface. In particular, both of these files are executed when you log in; first the .cshrc file and then the .login file. Similarly the commands in your .logout file are executed when you log out.

A common command to put in a .logout file is **clear**, which clears the screen. Typical commands to place in a .cshrc file are lines of the form
alias *shortname longcommandname*
which allows you to type *shortname* as a replacement for *longcommandname*.

One common use for the configuration files is to configure terminal settings to personal taste. For example, a user can choose which key is to be the "erase" character. We have configured accounts so that by default the BACKSPACE key will be the erase character, but users who prefer to use the DELETE key as the erase character can enter
stty erase '^?'
in their .cshrc to achieve this. Some users use the configuration files to change the operating system prompt. The default prompt is *machinename%* in tcsh.

If you are using an X-windowing interface, a key configuration file is ".Xdefaults". You can use this file to change such things as the presence or absence of scroll bars on your windows, the title of your windows, and the shape and colour of your cursor and numerous other things.

Another X-windowing configuration file is ".xsession". Commands in this file are executed every time you log in directly to *coxeter* using a windowing interface, before those from your ".login" are executed. Some people use this file to change the font or position of the initial window which appears after login. The .xsession file is the one which specifies which window manager you are using. The various window managers also have configuration files. For example, if your .xsession file specifies that you are using mwm (respectively twm) then your .mwmrc (respectively .twmrc) could be used to change the items which appear in the menus brought up by mouse presses.

Emacs has a configuration file called ".emacs" whose entries are executed every time the user starts *Emacs*.

The configuration file ".mailrc" is described in Chapter III.

11. Using Remote Machines

Having logged in to one UNIX machine, you may be able to access an account you have on another UNIX machine with the command **slogin** *machinename.* This will normally produce a password prompt from the other machine. If your login name on the remote machine is different from that on the machine you are using, you need to use the syntax **slogin** *remoteloginname@machinename.* The first time you use slogin to connect to a given machine, it will present RSA key fingerprint for the machine it thinks you have specified, and ask you to verify that you wish to connect. Under normal circumstances you should not be asked to verify the fingerprint of that machine again.

In order to run X on the remote machine your DISPLAY variable must be propagated. The command **slogin -X** *machinename* will automatically do this. If you have logged in through some method which has not automatically propagated your DISPLAY variable, you can set it manually with the command **setenv DISPLAY** *displayname* where "displayname" should be whatever response you get from **echo $DISPLAY** on the local machine you are using.

slogin has the advantage that the text you type is transmitted over the internet in encypted form. If the machine you (or perhaps your visitor) is trying to access is not accepting connections from either **slogin** or **rlogin** you are likely to receive the response "`Service unavailable`" or "`Unknown host` *machinename*" or some other error message. In this case the command "**telnet** *machinename*" might be successful. However, rlogin and telnet fail to employ encryption, so they should be avoided.

You can copy files to or from a "friendly" account (e.g. your account on another math server) by using the syntax **scp** *filename1 remotemachine:filename2* or **scp** *remotemachine:filename1 filename2* respectively. If your login name is different on the remote machine you need to use "user@remotemachine".

The older protocol, **ftp** *machinename* might also work, but like rlogin and telnet, it passes unencrypted information (including your password) over the internet, so is to be avoided. To learn more about **slogin** and **scp** type `scpinfo` at the *coxeter* prompt.

Within the department, "math" is a synonym for "coxeter.math". So **slogin math** will perform an slogin to *coxeter*.

If you are trying to connect to *coxeter* from a UNIX machine outside the University of Toronto, use the syntax **slogin math.toronto.edu** (or
slogin *loginname*@**math.toronto.edu** if your login name is different on the other machine). If your remote machine is running Microsoft Windows or is a Macintosh running a pre-OSX version of MacOS, you should find an slogin "client" for that operating system. For information on obtaining and downloading such clients, browse the URL `http://www.math.toronto.edu/help`. An slogin session can also be obtained using a Web-browser running Java, by clicking on the "Mindterm" button on that help page. Each time you do this, it downloads an slogin client into your browser-session, so is good for one-time-only connections. If you are connecting from a site you plan to use often, it would be advisable to download a client onto that machine, as above.

12. Local Variations

We have introduced a number of our own commands which are described in this section. Each machine on the network has a name. For example, some PC's in the computer room are called "*grad1*", "*grad2*", and "*grad3*". In order to display pictures (e.g. graphical output from *Mathematica* or TEX previews) on an X-windowing system your environment variable "`DISPLAY`" must have the value *name***:0**, where *name* is the name of the X display you are using. If you log on through an X-windowing system, your DISPLAY variable is automatically set correctly. One of the advantages of **slogin -X** is that it passes on the value of the DISPLAY variable to the second machine. However if you instead use the **rlogin** command, your DISPLAY variable will not be set on the second machine.

To print a text file on the default laser printer, (in BA 6290A, the mailroom), type **Print** *filename*. The corresponding command for the laser printer in the computer room (BA 6200) is **Print -Plw3** *filename*. Put the following into your ".cshrc" file (as described in Section 10) if you'd like to make the printer in the computer room your default printer: **setenv PRINTER lw3** and if you would like to make single-sided printing as your default, put: **setenv DUPLEXMODE 1**. Please note that the old **lpr** command does not work any more. To preview what will be printed on your display type **Print -d** *filename*.

The weekly seminar list can be viewed by entering **seminars** or **finger seminars**.

To see the list of software available on the various machines type **unixmachines**.

III. Mail

Please see: `http://www.math.toronto.edu/cms/email-and-spam-filtering/` for more up-to-date information about mail in our department. Below is the older documentation that we had.

pine is available for accessing mail. It has a primitive graphical user interface (the mouse is not used) but its basic menus are fairly self-explanatory. Type **man pine** for more info. Other mail-reading programs such as **mutt** have also been installed. **thunderbird** is a modern GUI based mail client that is recommended.

The mail system also allows for automatic forwarding of your mail from one address to another and various other automated features such as "vacation" messages. Type **vacation_email** for more help. One such automated feature is obtained through the powerful program "procmail" which nowadays is often used to automatically discard some pieces of mail which you regard as "junk mail", also known as "spam". The program is powerful enough to filter your mail based on almost any well-defined criteria you might wish to specify, although coming up with an algorithm to define what you regard as *spam* may not be easy. Using the more powerful features of *procmail* might be complicated but the **spamfiltering** command does most of what people usually want. **maillistedit** and **forwardedit** also can be useful for managing email.

Note that it might not be obvious who the true sender is of a message that you receive by email. The reliable line in the message header is the "Received from:" line, saying from which machine *coxeter* received the message. Other lines, including "From" lines, can say whatever the sender wants. Sometimes the forgers use local departmental addresses in the "From" lines. If you use Microsoft Windows programs to access your email please remember to not open attachments. Type "virus.warnings" for more information.

If you see a message from the `MAILER-DAEMON` that begins something like
`--- The Transcript of the session follows ---`
...
`550 john...  User unknown`
this means you tried to e-mail to the address `john` but that is not a valid e-mail address. A similar message is printed for invalid hostnames.

IV. Emacs

Emacs is an editor. That means it is a program for creating or changing the contents of text files. To start *Emacs*, type   **emacs** *filename*. If a file named *filename* exists it will be read into memory for editing; if not, a new empty file named *filename* will be created. The name of the file appears near the bottom of your window on the second last line. As much of your file as will fit appears above this. The last line is reserved for such things as messages from *Emacs* and instructions to *Emacs*. Somewhere within the main portion of the window will be a filled rectangle about the size of one letter. This represents the insertion point, hereafter known simply as *the point*. Characters you type are inserted into the window at the point, pushing any characters after the point further to the right. Pressing DEL deletes the character to the left of the point. Strictly speaking, the point is located at the left edge of the rectangle so it is always between characters.

The contents of the window which you see on the screen exist only in the computer's memory. To make a permanent copy of those contents you must "save" to the computer's disk. (See the 'save' command below.) Saving replaces the old version of the file on the disk with the new version. If you exit *Emacs* without saving or if some external event such as a system crash causes your *Emacs* session to be aborted, all record of what was on the screen is lost.

In addition to inserting characters at the point, you can give commands to *Emacs* to do things such as move the point to a different location, save a file, read in a new file, etc. There are in fact hundreds of commands that *Emacs* understands, although knowledge of a few of the most basic ones is all that is essential.

Running Emacs from an X-windowing system

Emacs recognizes two keys as being special and calls them the *Control Key* and the *Meta Key*. Commands are issued by typing special sequences of keys beginning with one of these two keys. The control key is marked "Ctrl" and the meta key is marked "Alt". The notation 'C-x' means to hold down the control key while typing the letter x. Similarly, 'M-z' means hold down the meta key while typing z. For example,   **C-c b**   means hold down the control key while typing c, followed by releasing the control key and typing b. A summary of the most useful commands is given below. In addition to these keystroke commands there are a few other ways of issuing commands to *Emacs*. For example, one can move the point simply by pressing the left mouse button at the desired location. Commands can also be issued by name; one types   **M-x**   followed by the name of the command to do this. One of the commands is a help command which gives details on other commands. It can also be used to bring up the *Emacs* tutorial.

The simplest way to use *Emacs* is just to press the mouse where you want the point to be and then type or delete characters. You can move longer distances with the   **C-v** and   **M-v**   keys. The "PageUp" and "PageDown" keys can also be used on some of the X-windowing systems to move these longer distances. It is useful however to know a few of the more powerful commands for operations such as moving segments of text from one place to another.

14

There follows a list of the most useful *Emacs* commands. The function of most of these commands is self-explanatory; others are explained below. A more extensive list is on the "GNU Emacs Reference Card" which can be obtained from Marco. There is a copy of the "GNU Emacs Manual" containing complete details of all *Emacs* commands in the main office. The manual is also accessible from within *Emacs* by typing **M-x info**.

<div align="center">Basic Emacs Commands</div>

C-x C-c	exit *Emacs*
C-x C-s	save file to disk
C-h	get help
C-g	abort partially typed or executing command
C-x u	undo last change
C-s	search (see below)
C-x C-f *filename*	read in (or create) a file for editing
C-b or ←	move backward one character
C-f or →	move forward one character
C-p or ↑	move backward one line
C-n or ↓	move forward one line
C-a	move to beginning of line
C-e	move to end of line
C-v (or **PageDown**)	move forward one screenful
M-v (or **PageUp**)	move backward one screenful
C-k	kill to end of line
C-y	insert most recent kill
C-SPC or **C-@**	set mark (see below)
C-x i *filename*	insert the contents of *filename* at point
C-w	kill region (see below)

In addition to the point, *Emacs* recognizes another special location in each file called *the mark*. The mark is used among other things for selecting a segment of text called *the region*, which is defined as the text lying between the point and the mark. Some of the more powerful *Emacs* commands operate on the region. You select a segment of text as the region by setting the mark at one end of it with **C-SPC** and then moving the point to the other end. (**SPC** denotes the space bar). *Emacs* distinguishes between "deleting" and "killing". Some commands (for example pressing the delete key) delete text, but others such as **C-k** delete the text while placing it in the *kill ring*. Text placed in the kill ring can be recovered using the **C-y** command. As an example, you could move a segment of text from one place to another with the following sequence:

  1 Move point to one end of text to be moved
  2 **C-SPC** (sets mark at this location)
  3 Move point to other end of text to be moved
  4 **C-w** (deletes text and places it in kill ring)
  5 Move point to desired location of text
  6 **C-y** (inserts text from kill ring)

Note: Unlike the point, there is no visible indication in the file as to the location of the mark unless one uses "transient-mark-mode" on an X-windowing system. Otherwise one generally knows where it is by deliberately setting it at the desired location immediately before calling a command such as **C-w** which uses it. You may use **C-x C-x** to exchange the point and mark in order to see where they are.

To search for the string *str*, type **C-s** *str* **RET** which will go to the first location after the point containing *str*. (**RET** denotes the RETURN or ENTER key). To repeat the search for the next location containing the same string, type **C-s C-s**. Repeatedly typing **C-s** continues searching for *str*; use **RET** to terminate the searching.

Running Emacs From an ASCII terminal

X-windowing systems have several features which are not available on "standard" ASCII terminals. It is possible to use *Emacs* without X, but more reliance must be placed on the C-*key* and M-*key* commands since other methods (such as mouse, PageUp key, arrow keys, etc.) may not be available. The biggest difference is that without X, the ALT key no longer functions as *Emacs'* meta key. Instead one enters, for example, 'M-F' by pressing *and releasing* the ESC key followed by pressing *F*. (If you press and hold down the ESC key the way you do with the Ctrl key or a real meta key it will "auto-repeat". Should you accidentally do this, you can get back to where you were by using the **C-g** abort command.)

V. T_EX

coxeter has our most recent versions of T_EX, L^AT_EX, $\mathcal{AMS}$-T_EX, and $\mathcal{AMS}$-L^AT_EX. METAFONT, for creating arbitrary size fonts for documents is also available. To prepare a document, use an editor such as *emacs* or *vi* and label your file *filename*.**tex**. If it is to be an $\mathcal{AMS}$-L^AT_EX file, use L^AT_EX with an appropriate option (such as "amstex") which will allow you to use NFSS (the new font selection scheme). To run T_EX or $\mathcal{AMS}$-T_EX, type **tex** *filename*, or **amstex** *filename*; for L^AT_EX and $\mathcal{AMS}$-L^AT_EX, use **latex** *filename*. This should produce a new file called "*filename.dvi*". You can print this file on the laser printer in the mailroom (BA 6290A) by typing **Print** *filename*.**dvi**. To print on the laser printer in the computer room (BA 6200), use **Print -Plw3** *filename*.**dvi**. You can also type **Print -1** *filename*.**dvi** if you want to have a one-sided printout. This (usually) works with both printers. It is also possible to print a portion of the file by using the syntax illustrated in the following examples:

Print -p 4 *filename*.**dvi**   to print page 4
Print -p 3-7 *filename*.**dvi**   to print pages 3 through 7
Print -p 2,6,8 *filename*.**dvi**   to print pages 2, 6, and 8

For an online summary of options available, simply type **Print** on its own at the shell prompt.

Before printing your document, it is recommended that you first preview it on the screen to ensure that it looks the way you want and not waste paper. The best way to

do this is through the **xdvi** command from an X-windowing system. Type **xdvi** *filename* to get a preview in which you can see most of the page at once at reduced size, or **xdvi -s 2** *filename* for a full size preview showing only a portion of a page at a time. In the latter case, you can use the mouse and scrollbars to bring different portions of the page into view. You can also use the keyboard commands "d", "u", "r" and "l" to move up, down, right and left. The simplified abbreviation **dvi** *filename* will bring up the preview window (at reduced size) near the upper left corner of the screen. A rudimentary preview can also be obtained from an ASCII terminal, using the command **dvi2tty** *filename*. You will not be able to see any detail from this type of preview, but it is sufficient to see the general spacing of the material on the page, and thus to determine if there are gross errors.

Manuals for TeX, LaTeX, $\mathcal{AMS}$-TeX, and METAFONT are in the main office. Many members of our department including our office staff are very expert in the use of TeX and help for making tricky constructions is usually available.

VI. Mathematica

When on *sphere* you can start a *Mathematica* session from the shell command line, by typing **math** or **mathematica** (for a notebook interface)

Although the details of the interface you will see in each case are slightly different, the structure of *Mathematica* calculations is the same. You type something; then *Mathematica* processes it, and returns a result, prefixed by `Out[]=`. If you are within a *Mathematica* session started from the shell, you will receive an `In[]:=` prompt from *Mathematica*. If you are within *Emacs*, you will not receive these prompts, but instead will see the message `Done Out[]` at the bottom of the screen when *Mathematica* is ready for your next input. To leave *Mathematica*, enter the command **Quit**.

We can present only a brief introduction to *Mathematica* here, just enough to get you started. The reference for *Mathematica* is: **Mathematica: A System for Doing Mathematics by Computer** by Stephen Wolfram. Copies of this book are available in the main office; Chapter 1, "A Practical Introduction to Mathematica", is highly recommended reading. In addition, *Mathematica* provides on-line help and documentation features.

You can do arithmetic using *Mathematica*, by typing in arithmetic expressions:

`In[1]:= 2 + 5`
`Out[1]= 7`

The symbols $+$, $*$, $/$, $-$, $\hat{ }$ stand for plus, times, division, minus, power. A space can also be used to denote multiplication:

`In[2]:= 2 5`
`Out[2]= 10`

The precedence of the operations is the same as in ordinary arithmetic; parentheses can be used to control order of evaluation.

Mathematica is different from an electronic calculator in that it will try to give you exact results, not numerical approximations, unless a number is given with a decimal point. For example:

```
In[3]:= 1/3 + 2/7
```
$$\text{Out[3]} = \frac{13}{21}$$

If you happen to want the numerical approximation instead, surround the expression with "N[...]":

```
In[4]:= N[1/3 + 2/7]
Out[4]= 0.619048
```

Mathematica does symbolic as well as numerical calculations. That means that it can handle algebraic formulas as well as numbers:

```
In[5]:= 3x − x + 2
Out[5]= 2 + 2x
```

Mathematica's symbolic capabilities include substituting variables, for example:

```
In[6]:= y = (a + b)
Out[6]= a + b
In[7]:= Expand[y^2]
```
$$\text{Out[7]} = a^2 + 2\ a\ b + b^2$$
```
In[8]:= b = 5 + c
Out[8]= 5 + c
In[9]:= Expand[y^2]
```
$$\text{Out[9]} = 25 + 10\ a + a^2 + 10\ c + 2\ a\ c\ + c^2$$

<u>Note</u>: $a\ b$ is interpreted as the product of the variables a and b, whereas ab would be a variable named "ab".

Mathematica has over 600 built-in functions. The names of these functions all begin with capital letters, and their arguments are enclosed in square brackets.

Some familiar ones are:

Sqrt[x]	square root
Exp[x]	exponential
Log[x]	natural logarithm
Log[b,x]	logarithm to base b
Sin[x], **Cos[x]**, **Tan[x]**	trig functions (arguments in radians)
Factorial[n] or **n!**	factorial
Abs[x]	absolute value
Mod[n,m]	n modulo m
Random[]	pseudorandom number between 0 and 1
FactorInteger[n]	prime factors of n

These functions can be applied to symbolic arguments, returning symbolic results:

```
In[10]:= Sqrt[(1 + x^2)^4]
```
$$\text{Out[10]} = (1 + x^2)^2$$

Mathematica has common mathematical constants built in:

Pi π
E e
Degree $\pi/180$
I i (Sqrt[-1])
Infinity ∞

One of the most useful features of *Mathematica* is its ability to manipulate lists. A list can be created by giving its elements between braces separated by commas:

`In[11]:=` $mylist = \{5, \ x, \ 3, \ abc, \ x\}$
`Out[11]=` $\{5, \ x, \ 3, \ abc, \ x\}$

Individual elements of lists can be referred to by their position within the list:

`In[12]:=` $mylist[[2]] + mylist[[4]]$
`Out[12]=` $abc + x$

There are also many built-in functions which produce lists:

`In[12]:=` Table[$i\hat{\ }2, \ \{i, \ 2, \ 7\}$]
`Out[12]=` $\{4, \ 9, \ 16, \ 25, \ 36, \ 49\}$

A matrix is a list whose elements are themselves lists having a common length.

`In[13]:=` $m = \{\{a, \ 7\}, \ \{3, \ x\}\}$
`Out[13]=` $\{\{a, \ 7\}, \ \{3, \ x\}\}$
`In[14]:=` Det[m]
`Out[14]=` $-21 + a \ x$

In addition to numerical and symbolic capabilities, *Mathematica* handles other standard mathematical operations. For example:

D[f [x]]   derivative of $f(x)$
Det[m]   determinant of the square matrix m
Solve[$x\hat{\ }2 - 3x + 2 == 0$]   solve the equation $x^2 - 3x + 2 = 0$

Mathematica also has graphics capability. **Plot[Sin[x], {x, 0, 2 Pi}]**   will plot the graph of $\sin(x)$ in the range $0 <= x <= 2\pi$.

Mathematica is a programming language, incorporating features from C, APL, PROLOG, and LISP. You can define a variable:

`In[15]:=` $x = 7$
`Out[15]=` 7

and clear its value:

`In[16]:=` Clear[x]

You can define your own functions by giving a sequence of *Mathematica* statements using the syntax:

myfunction := (1*st statement*; 2*nd statement*; ... ; *last statement*)

The value of a function is the value of the last statement executed. Function arguments are placed in square brackets after the function name and are followed by an underscore. For example, the following would define a function f such that $f[n]$ is the coefficient of x^2 in the polynomial $(1 + x)^n$:

`In[17]:=` $f[n\_] :=$ (*poly* $= (1 + x)\hat{\ }n$; Coefficient[*poly*, $x, 2$])

```
In[18]:= f[4]
Out[18]= 6
In[19]:= f[7]
Out[19]= 21
```

When defining your own variables and functions, it is best to choose names that begin with lower case letters, in order to avoid conflict with *Mathematica*'s built-in functions.

If you want to suppress the output that *Mathematica* would otherwise generate in response to your input, append ";" to the expression you type in:

```
In[20]:= x = y + 7;
```

You can interrupt a computation in progress by typing *Ctrl*-**c** if you are using *Mathematica* from the UNIX command line, or by typing **C-c C-c** if running *Mathematica* from within *Emacs*. *Mathematica* will give you the option to abort at this point. Sometimes the abort doesn't work, however, and you have no alternative then but to quit your session and start a new one. (This is a problem with the design of *Mathematica* — in order to speed up computations, *Mathematica* does not pay close attention to abort requests.)

In order to see the on-line documentation for a function called *Name*, type:

```
In[21]:=?Name
```

You can get more extensive documentation by typing:

```
In[22]:=??Name
```

In addition to the functions that are built in to *Mathematica*, functions may be defined in files external to *Mathematica*, called packages. Typically, these files have names that end in ".m". In order to have access to the functions in such a file, you read it in during your *Mathematica* session. Thus:

```
In[23]:= <<MyPackage.m
```

This would make the functions defined in *MyPackage.m* available in your current file.