6 August 2019

LOCAL GUIDE

to accompany the use of

COMPUTATION AND PROBLEM SOLVING IN UNDERGRADUATE PHYSICS

at

Lawrence University

Template by:

DAVID M. COOK

Department of Physics
Lawrence University
Appleton, Wisconsin 54911

Adaptation for *Lawrence University by*:

DAVID M. COOK

Department of Physics
Lawrence University
Appleton, WI 54911

PERMISSIONS

To facilitate adaptation and use of this *Local Guide* at places beyond Lawrence University, the license under which this document and associated files are distributed grants you, David M. Cook, permission to edit the Lawrence template as appropriate to your department at your institution and to reproduce sufficient copies for your use in your instructional program at Lawrence University, provided

(a) that you include in your copies the title page with the license statement and this statement of permission as edited to identify you and your institution, and

(b) that you make copies available to your students at a cost not exceeding the cost of production (including any markup your bookstore may add for handling the distribution).

Preface and Acknowledgements

This supplement to *Computation and Problem Solving in Undergraduate Physics* (*CPSUP*) by David M. Cook provides information about the environment in the Computational Physics Laboratory (CPL) at Lawrence University. At various points in *CPSUP*, the reader is referred to this local guide for specific information about the hardware and software available in the CPL.

In the summer and fall of 2012, the hardware in the CPL was updated by the creation of a virtual server in the Lawrence library and the replacement of workstations serving the Department of Physics since 2005 with twelve new HP workstations. Upfront, the author wishes to acknowledge the assistance during the summer of 2005 of Claire Weiss, LU '07 and Erik Garbacik, LU '08, who explored the Linux operating system on the first HP platform, helped in configuring the HP computers for most convenient incorporation into the CPL, and generated initial drafts of major portions of the HP and Linux components in the revised and updated version of this *Local Guide*. In particular, Ms. Weiss and Mr. Garbacik deserve special recognition for suggesting and then drafting the all new Section 16, which contains tips from experienced student users of the resources of the CPL.

Further edits, though small by comparison, were made in December of 2006. Additional edits were made in the fall of 2007, mainly to reflect the upgrade of the HP computers from the Fedora Core 4 implementation of Linux to the Fedora 7 implementation. The next printing (fall, 2012) updated the text to reflect new HP workstations, which utilize the Fedora 17 implementation of Linux, and deleted all references to the now abandoned SGI workstations. Finally, this printing (fall 2018) makes further updates in several spots and, in particular, adds information about PYTHON, OCTAVE, and MAXIMA, several of which have recently been installed in the CPL.

David M. Cook
Appleton, Wisconsin
6 August 2019

Table of Contents

Local Guide

Computational Physics Laboratory
Department of Physics – Lawrence University

As listed in the table in Appendix A, the Lawrence University Computational Physics Laboratory (CPL) is equipped with twelve six-processor Hewlett-Packard (HP) Z620 workstations, a virtual server located with other Lawrence University servers, an HP P4015x monochrome laserjet PostScript printer, and an HP 4700dn color laserjet PostScript printer. This networked hardware and associated software provide sophisticated computational and graphical resources in support of pedagogic and scholarly pursuits in physics. This guide introduces some of the features of these devices and describes (briefly) some of the available software. Fuller detail about the software and numerous applications of its use in physics can be found in *Computation and Problem Solving in Undergraduate Physics* (*CPSUP*) by David M. Cook and in several (now outdated) locally prepared documents designated by codes beginning CPL-, e.g. CPL-121, and collected in notebooks in the CPL library.

Much of the software on every device in the CPL is proprietary and subject to the provisions both of the United States copyright laws and of license agreements between Lawrence and the vendors of the software. In most cases, the licenses permit simultaneous use on all of the devices in the CPL; in all cases, the licenses limit use to projects and activities at Lawrence University and prohibit copying of the software, except for purposes of system maintenance and backup. All users of all devices must be constantly mindful of the proprietary nature of much of the software in the CPL and must abide by the restrictions imposed by the copyright laws and by the license agreements. Specific information about these restrictions for software available in the CPL is included later in this guide.

1 Important Preliminaries

As a preamble to describing ways to work with the resources of the CPL and, in particular, introducing some of the features of the operating system, we present a collection of miscellaneous facts about the system. These items are presented in no particular order. Further, you may find it valuable to review them once in a while during your first month of working in the CPL.

1. The HP workstations use the Fedora 17 implementation of the Linux operating system. You will have to learn some (but not much) UNIX in order to use these workstations. A brief introduction to UNIX appears in Section 2. Additional information will be found, for example, in Kernighan and Pike,[1] in Sobell, and in voluminous documentation in the form of `man`-pages (manual pages), in-program on-line help, and other on-line resources.

2. In (almost) all contexts, *UNIX is case-sensitive*. The files `Example.f` and `example.f`, for example, are *different* files.

3. When a file being written has the same name as an existing file, the new *overwrites* the old and the old is *lost*—only sometimes with a warning and request for confirmation. *Beware!*

4. ⟨CONTROL-C⟩ terminates execution of the current job, returning control to the operating system.

5. ⟨CONTROL-D⟩ is used as an end-of-file character. For example, it terminates messages typed into the UNIX mail program, and it can be used as a quick way to close a shell.

6. ⟨CONTROL-Z⟩ suspends a job and leaves it hanging (but *not* running) in the background.[2]

[1] References are more fully identified in Section 18.
[2] The command `fg` brings the job back to the foreground.

7. Full filenames in UNIX start with a '`/`', continue with the names of whatever directories define the path, and conclude with the name of the file itself, e.g., `/home/cookd/`*FileName*. Wild cards include the asterisk '*' to stand for any number (including zero) of any characters and the question mark '?' to stand for any single character.

8. Each user's private home directory resides on a disk physically connected to the file server, node NEWTON, and is identified by the path `/home/`*YourUserName*. This disk is mounted on all nodes so users have access to the *same* directory no matter which physical satellite machine they choose to use.

9. Filenames and directory names in UNIX can be constructed out of (upper and lower) case letters, numeric digits, and some special characters, including '`.`' (dot), '`_`' (underscore), '%' (percent sign), and '`-`' (hyphen) but *not* including '`*`' (asterisk), '`?`' (question mark), and '`/`' (forward slash). (As described in item 7 above, these last three characters have other special meanings when they occur in filenames.) Filenames can also contain the space character, but all such file names must be enclosed in (single or double) quotation marks when they appear in commands. Use of spaces in file names is imprudent and, in some contexts, inconvenient.

   Note, in particular, that the character '`.`' (dot) in a UNIX filename is just another legal character, and a UNIX filename can have any number of dots. From the perspective of the *operating system*, the dot does *not* separate the main part of a filename from a file type, though the *user* can certainly view the dot in that way. Only rarely will UNIX programs assume a default file type when none is specified.

10. The mouse has three buttons, which we will denote ML, MM, and MR (mouse left, mouse middle, mouse right) in this guide. On the HP workstations, MM plays a second role as a scroll wheel.

11. Backup follows two independent patterns. Daily at 4:00 AM, a snapshot on disk of the `/home` directory (all user files) and several other files not part of the base operating system on the server is made, with one copy stored near the server and a second copy (replication) sent to another building on campus. These snapshots are stored for ten days and can be *very* quickly reinstalled in the event of a need. In addition, a full backup of all files on the server, including the full operating system, is made every Saturday at 3:00 AM and differential backups[3] on tape are made at 3:00 AM Sunday through Friday. Thus, at any time, we can recover files as they were the previous morning at 3:00 AM or 4:00 AM.[4]

12. UNIX files can be given a variety of permissions that control who has what kinds of access to each file. Basically, files are owned by a particular user who will be a member of at least one group as well. Various *modes* allow control over read (`r`), write (`w`), and execute (`x`) access separately for the user (`u`), for the group (`g`), and for others (`o`, i.e., everybody). Given the default setting of the *permissions mask*, files created in a *Shell* window (see Section 3) are created with `rwx` access for the user alone, though `x` will be there only for executable files. Only if the user specifically changes those permissions will members of the group or others have any access to the file at all.

13. By default, for a particular window to be *active* (i.e., to be the focus of input typed on the keyboard), the cursor must be positioned somewhere in the visible region of that window. An active window need not, however, be in the foreground. To bring a window to the foreground on an HP workstation (and make it active), click ML anywhere in the window. Further, repeated use of ⟨ALT-ESC⟩ will cycle through open windows, bringing each in turn to the foreground (though the cursor will still have to be in the window to make it active).

[3] A differential backup stores all files that have changed since the most immediately past full backup.

[4] If you have made major changes in files or, for some other reason, are feeling particularly vulnerable, and the next backup is uncomfortably far off, you can use one of the USB ports on the HP workstations to write the files to a flash drive. See Section 14 for more information about saving data off-line.

2 A First Session

In this section, we guide you through a very quick, first session on a workstation in the CPL, our aim being to help you develop sufficient familiarity with the underlying operating system so that you will be able to use its features automatically and comfortably in all subsequent work. Basically, any session will involve logging in, doing something (perhaps several things), and logging out.

Some aspects of the user interface (an X-window environment) on the workstations in the CPL look and feel like a Microsoft Windows user interface; other aspects look and feel like a Macintosh user interface. We here assume you have at least a modicum of familiarity with at least one of those interfaces. In the remainder of this section, we describe how to log in, how to work in a *Shell* (command line) window, how to use some of the the features of the Graphical User Interface (GUI) or *Desktop*, and how to log out.

2.1 Logging In and Creating a *Shell* Window

The first step in using a workstation in the CPL involves logging in and creating a *Shell* window in which you can enter commands to the operating system. To log in, proceed as follows:

1. If the screen is dark, move the mouse to wake the monitor out of its power-saving mode. In addition to a portrait of the physicist after whom the node is named, a list of users who have previously logged in on your machine will appear in a few seconds. (If it doesn't, you may have to turn the monitor on.)

2. If your name does not appear in the list on the screen (which will be the case at your *first* log in on a given machine), then

   - Click ML on the item 'Not listed',
   - Type your user name[5] in the the resulting text entry box labeled 'Username',
   - Type the ⟨ENTER⟩ key—*yes the ⟨ENTER⟩ key, not the ⟨TAB⟩ key*,
   - Type your current password[6] in the resulting text entry box labeled 'Password', and
   - Click ML on 'Sign In'

   else

   - Click ML on your name,
   - Enter your password in the screen that comes up, and
   - Click ML on 'Sign In'.

Sooner or later, the workstation screen, which in its entirety is referred to as the (GNOME) *Desktop*, will display

- The *top panel* across the top of the screen and containing, by default, an Applications menu and a Places menu at its left end, a clock displaying the date and the time in the middle,[7] and a couple of icons and a menu titled with your name at its right end.

[5] Unless there are conflicts, your username consists of your last name (truncated to seven characters if necessary) followed by your first initial, all in lower case letters. In any case, it is the same as your username on other University computers. You user name will be displayed as you type it.

[6] For a newly created account, your password is your last name (truncated to seven characters is necessary) followed by the numeral '1', followed—if necessary—by enough of the string '23456' to make a total of eight characters. For security reasons, passwords are not displayed as you type them. You will not be forced to change your password, but you should do so as soon as convenient. See the instructions in item 1 in Section 2.2.

[7] If you click ML on this clock, a calendar for the current month will pop up. Left and right "arrows" on the month and year allow shifting to different months and years and a button labeled 'Edit' brings up a screen in which you can customize various features of this display. The calendar can be closed by clicking ML on the phrase in the top panel. Clicking MR on the clock in the toolbar brings up a different menu offering additional options.

- The (probably nearly blank) *bottom panel* across the bottom of the screen. Once you have open windows on your desktop, an icon/button for each will appear in this lower tool bar. Initially, the bottom panel probably contains at its far right end only a small graphical depiction of the various work spaces in their current configurations.
- Three icons (Computer, home, and Trash) vertically along the left edge.
- *Possibly* a *Shell* window, in which commands can be entered.

As a whole, the desktop gives a graphical user interface (GUI) to many functions of the system, some of which we will explore in this guide.

We also include in this section a brief exploration of a very few of the features of the HP desktop, ending with the creation of a *Shell* window so you are prepared for the UNIX tutorial in Section 2.2.

3. Open the Applications menu at the left end of the top panel.[8] The resulting menu contains several items. Those ending with a (right-pointing) greater-than sign themselves contain submenus, each of which can be "opened" and viewed simply by moving the cursor down the Applications menu to highlight the desired item.

4. Note the list of entries in each submenu of the Applications menu. In particular,

   - In the Accessories (sub)menu, the items 'Advanced Settings', 'Calculator', 'Emacs Text Editor', 'gedit Text Editor', 'Help', 'Screenshot', and 'Search for Files ...'.
   - In the Graphics (sub)menu, the items 'EVINCE Document Viewer', 'Image Viewer', and 'Tgif'.
   - In the Internet (sub)menu, the items 'FireFox', 'FTP Client', and 'gFTP GUI Interface'.
   - In the Office (sub)menu, the items 'Adobe Reader 9', 'Dictionary', 'EVINCE Document Viewer', several links to components of LibreOffice (which emulates components of Microsoft Office in LINUX), and 'Xdvi previewer'.
   - In the System Tools (sub)menu, the items 'System Monitor', 'System Settings', and 'Terminal'.

5. Open the Places menu at the left end of the top panel and note the items 'Home Folder', 'Downloads', 'Computer', 'Search for Files ...' and 'Recent Documents'.

6. Open the YourName menu at the right end of the top panel and note, in particular, the items 'System Settings' and 'Log Out ...' in the resulting list.[9]

7. We have tried by default to set accounts so that the active window is the one in which the cursor is located, regardless of whether that window is in front of all others on the desktop. To confirm that option, migrate to Applications $\longrightarrow$ Accessories $\longrightarrow$ Advanced Settings $\longrightarrow$ Windows. In the box labeled 'Window focus mode', make sure the selected option is 'Mouse'. Also make sure that the boxes labeled 'Action on title bar double-click', 'Action on title bar middle-click; and 'Action on title bar right click' are set respectively to 'Toggle Maximize', 'Lower', and 'Menu'. These actions will verify that the defaults assumed in the remainder of Section 2 are in fact established (and establish them if they aren't). Subsequently, you can if you wish set these boxes to other values. Close the *Advanced Settings* window.[10] This operation removes the window from the desktop altogether.

[8] We shall use the phrase "open the ..." for an operation that involves clicking ML *once* on the corresponding icon or button.

[9] We are trying to find the way to remove the item 'Shut Down ...' from this menu. In the meantime, we urgently ask that you pretend it is not there. Shutting down the computer altogether is a violation of the rules of citizenship in the CPL. See Section 17.

[10] We shall use the phrase "close the ... window" for an operation that involves clicking ML on the button marked with the symbol '×' in the upper right corner of the window. Alternatively, in *many* windows (though not this one), you can close a window by selecting 'Close' or 'Close Window' from the File menu to which the icon in the far upper left of the window gives you access.

8. Select 'Terminal' from the SYSTEM TOOLS submenu of the APPLICATIONS menu in the top panel.[11]

9. Minimize the *Terminal* window just created.[12] This operation reduces the window to a labeled button in the bottom panel.

10. Reopen the window just minimized by clicking ML on its icon in the bottom panel.

11. Close the *Terminal* window.

12. Look in the top panel. *If you do not see an icon in the shape of a small screen and labeled 'Terminal' (when you move the cursor over the icon)*,

    - Open the SYSTEM TOOLS submenu of the APPLICATIONS menu,
    - Move the cursor to highlight the item 'Terminal' near—if not at—the bottom of the SYSTEM TOOLS menu,
    - Press *and hold* ML,
    - Move the cursor to a convenient location in the top panel, thereby dragging an icon to that toolbar, and
    - Release ML.

    This operation will place an icon in the top panel and greatly facilitate the opening of one or more *Shell* windows.[13]

13. Open a window for the entry of statements by clicking ML on the 'Terminal' icon in the top panel. Then,

    - Move the cursor into the title bar across the top of the window, press and hold ML, then drag the window to position it where you wish on the screen, and release ML.
    - Move the cursor into the title bar and double-click ML. The window will be expanded to fill the screen.
    - Click ML on the button in the upper right corner of the screen marked with □. The window will be returned to its original size and position.
    - Move the cursor into the title bar and depress MR. A pop-up menu will appear and, among other things, offer you the options to minimize, maximize, move, resize, or close the window.
    - Select 'Move', move the cursor to reposition the window, and click ML.

Your interface to the computer is now set for you to move on to the tutorial in Section 2.2. Statements to the Linux operating system may now be typed in this window. These statements will access the current *default directory*, which, right after you log in for the first time, will be your private home directory `/home/`*YourUserName*. (See the `cd` command described in Section 2.2 for the way to change the default directory.)[14]

[11] Normally, the phrase "select ..." will mean to click ML on the identified item. The phrase "select ... from the ... menu" describes a more complicated operation in which you obtain the desired item by (1) depressing (and holding) ML with the cursor on the appropriate icon or button, dragging the cursor to highlight the identified item in the resulting pop-up menu, and releasing ML or (2) clicking ML on the appropriate icon or button and then, in the pop-up menu, clicking ML on the appropriate icon or button. The phrase "select ... from the ... submenu in the ... menu" describes a still more complicated operation in which the menu and then the submenu must be opened in turn before ML can be clicked on the specified item in the submenu.

[12] We shall use the phrase "minimize the ... window" for an operation that involves clicking ML on the 'Minimize' button—the one with the underscore symbol—in the upper right corner of the window.

[13] Icons placed in the top panel can be removed by holding down the ALT key, moving the cursor onto the icon to be removed, depressing MR, highlighting the item 'Remove from panel', and releasing MR. This action only removes the icon from the toolbar; it does not delete the corresponding program from the system.

[14] During your first login, the number that appears at the end of the first prompt in every *Shell* window that you open will probably be 1. In subsequent logins, the number in the first prompt of each window is very subtly determined. A brief explanation is contained in Section 3.

2.2 Working in a Shell

We begin with the command line interface in a *Shell* window. Remember that, in the default environment, the cursor must be positioned somewhere in the visible portion of the *Shell* window for the window to be active, i.e., to accept input from the keyboard—though the window need not be in front of all other windows.[15] To explore quickly the manipulation of the environment, move the cursor into the *Shell* window and type the statements

> *Note that, if at any point you run out of time, you can easily interrupt your session. Simply skip to Section 2.4 for instructions on how to log out, and then log in again when you have time to return to the tutorial.*
>
> *Remember also that, by default, while you can type characters into a window that is not in front of all other windows, you cannot type characters into a window unless the cursor is positioned somewhere within the visible portion of that window.*

1. *Prompt* 1% `yppasswd`

   > *Depending on what you have done in previous logins, the number at the end of your first prompt in this session may not be 1. Whatever the starting number, it will be incremented by 1 with each new prompt.*
   >
   > *By local convention, the prompt for a UNIX statement on the workstations in the CPL has been set to be your username followed by an @ sign followed by the node name followed by the name of the default subdirectory followed by a number followed by a percent sign, e.g.,* `cookd@brahet CCLI 24%`*. Ask for help if you wish to set your own personal prompt to something else. In this guide, we have for brevity and generality used 'Prompt n%'.*

   This statement invokes a program that will allow you to change your password.[16] It asks first for your old password and then twice for the desired new password. As presently configured, your initial password is not flagged as expired, so you will not be *forced* to change it. Changing that initial password to something you can remember and no one else knows is nonetheless strongly recommended.[17]

2. *Prompt* 2% `cp /apps/CPSUP/lg/laplace.f.gz .`

   This statement copies the file `laplace.f.gz` *from* the public directory `/apps/CPSUP/lg` specified by the first argument *to* the current default directory, which here—and in many other contexts—can be abbreviated with the single character '.', specified by the second argument.[18] With this form, the file appears in your (default) directory with the *same* name that it had in the original directory. Alternatively, the final dot ('.') could be replaced by a desired *new* name for the file.

   Note, incidentally, that the `cp` command creates a copy that preserves whatever of the permissions possessed by the original file are allowed by the current setting of the permissions mask.

[15] Note the difference in the appearance both of the cursor and of the symbol after the system prompt when the cursor is in or, alternatively, out of the *Shell* window.

[16] The command `passwd` is more common for changing passwords. Note, however, that, when it can be done at all, a password created with `passwd` is local to a particular node. `yppasswd` changes a network password, which can then be used to log on to any node which doesn't have a local password. To avoid generating multiple passwords and the confusion which can result, use only `yppasswd`.

[17] Note that, in some some operating systems, passwords must have at least eight characters and contain at least one numeric character and at least one upper case character. We haven't yet determined whether these constraints apply in Fedora 17. Nonetheless, longer passwords with a mix of upper and lower case letters and numbers are most secure. Our operating system may provide judgments on the security of an entered password but may not actually prevent the use of such passwords.

[18] You will be presented with an error message if you leave off the concluding dot in this statement.

The copy will be given the time and date of its creation, not the time and date of the original file.[19]

3. *Prompt* 3% `gunzip laplace.f.gz`

   This statement executes the program `gunzip` with argument `laplace.f.gz`, "unzipping" the file `laplace.f.gz` from a compacted ("gzipped") form that is difficult to read and creating *in its place* the (larger) file `laplace.f`, which is a standard ASCII file that can be displayed as text on the terminal. *For now, do not delete the file* `laplace.f` *from your directory. We will use it later to illustrate other features of the system.*

4. *Prompt* 4% `ls`

   This statement requests a listing of the files in the current default directory (your home directory). Your logging in will have created directories named `Desktop`, `Pictures`, `Videos`, and several others, so this command will report several files and directories beyond `laplace.f`.

5. *Prompt* 5% `ls -al`

   This statement requests a more detailed listing of the files in the current default directory. Among other details for each file), it displays files and directories that were "hidden" from the first statement, i.e., files whose names *begin* with a dot (e.g., `.cshrc`, which—as described later—plays a role in establishing the details of your user environment and '.' and '`..`', which are system files that facilitate your reference to the current directory '.' and its parent directory '`..`').

   As illustrated here, many UNIX commands can be modified by the specification of options, which invariably follow the name of the command and are (almost always) introduced by a hyphen (often read 'dash'). In the statement '`ls -al`', `a` (for all files, including hidden files) and `l` (for a "long" directory) are some of the many options valid with the `ls` command.

   Now take a closer look at the information in the display on the screen. One line corresponds to each file. From left to right, each line shows the file type and permissions,[20] the number of links to the file, the owner, the group, the file size (in bytes), the date and time of last modification, and the file name.

6. *Prompt* 6% `chmod go+r,u+w laplace.f`

   This statement changes the permissions mode of the specified file by adding read (`r`) access for group (`g`) and others (`o`) and adding write access for the user (`u`, i.e., the owner).

7. *Prompt* 7% `ls -al`

   This statement will display the directory again so that you can verify the action of the `chmod` command. The more restrictive form '`ls -al laplace.f`' will display the information for only the one specified file.

8. *Prompt* 8% `cat laplace.f`

   This statement displays the contents of the specified *text* file on the screen.[21] Since it spits the entire file out without pause, a long file will almost certainly scroll by faster than any human can read it.

[19] Replacing `cp` with `cp -p` will preserve protections, dates, and times in the copy.

[20] This information is conveyed in a string of ten characters. The first character codes the file type (`-` for ordinary file, `d` for directory, `l` for link, etc.). The remaining nine characters are to be seen as three groups of three, each group conveying whether read, write, and execute permissions have been granted to the user, the group, and others, respectively.

[21] If the file contains a directory, `cat` will generate an error message. If, however, the file is a binary file, `cat` will produce gibberish on the screen; typing ⟨CONTROL-C⟩ should stop the display and return to a prompt from the operating system. (If this operation leaves the *Shell* window with a strange character set, you may have to close that window and open a fresh one.)

9. *Prompt* 9% `more laplace.f`

   This statement displays the contents of the specified *text* file on the screen, one screen full at a time.[22] Pressing the *space bar* will advance the display to the next screen, and typing a (lower case) '`q`' will abort the display of the rest of the file and return to the prompt from the operating system. Note that each screen full contains a (highlighted) last line indicating, in addition to the the word 'More', the percentage of the file so far displayed.

10. *Prompt* 10% `ls -lR /apps/CPSUP`

    This statement lists all files in the identified directory, including—option `R`, for recursive—not only the directory itself but all subdirectories descending from the identified directory. The listing occupies more than one screen full, but the entire directory is displayed in a non-stop scroll. You will surely have difficulty reading it as it scrolls by.

    More specifically, the identified directory is the head directory for many of the files referred to in *CPSUP*. Throughout *CPSUP*, the subdirectory `/apps/CPSUP` is referred to with the symbol `$HEAD`. Any references in *CPSUP* to files in the directory `$HEAD` should at Lawrence be understood to be references to files in the directory `/apps/CPSUP`. A full listing of all of the abbreviations used in *CPSUP* and their translations to the Lawrence environment is included in Appendix B.

11. *Prompt* 11% `ls -lR /apps/CPSUP | more`

    This statement is actually *two* statements, in which the output of the first statement (the same listing displayed by the previous example) is *piped* (conveyed by the vertical bar) as input to the second statement. The result is a page-by-page display on the screen.

12. *Prompt* 12% `ls -lR /apps/CPSUP > CPSUP.txt`

    This statement illustrates redirection of output. In this case, the output of the `ls` command is written not to the screen but to the file `CPSUP.txt` in the default directory. That file—a standard ASCII file—can then be examined with `more` or `cat` or, perhaps more usefully, directly printed (Section 5). *For now, do not delete the file* `CPSUP.txt` *from your directory. We will use it later to illustrate other features of the system.*

13. *Prompt* 13% `cat .cshrc | grep -n '#' > comments.txt`

    This statement pipes (`|`) the output of the `cat` command—i.e., the entire file `.cshrc`—as input to the `grep` command, which, with the specified option and argument, scans its input for occurrences of the special character '`#`' (which marks comments in the file) and, through redirection of output (`>`), writes only those lines containing that character (the comments) to the file named `comments.txt`, preceding each—option `n`—with the corresponding line number in the original (full) file. The file `comments.txt` can, of course, be examined with `more`, `cat`, or a text editor.

14. *Prompt* 14% `gedit comments.txt`

    This statement will open the file `comments.txt` just created and display it in the recommended text editor. We will discuss the actual use of `gedit` for editing a text file in Section 4.1. For now, use the *gedit* window—which will be labeled with the name of the file you are editing—to explore moving and re-sizing windows.

    - Move the cursor into the title bar (i.e., the bar across the top of the window in which the title appears; the cursor will retain the form of an arrow), press and hold ML, drag the window to a new location, and release ML.

[22] See footnote 21.

- Move the cursor to one of the borders (i.e., the narrow band around the entire window) but not too close to a corner. The cursor will change into a line parallel to the border with an arrow pointing into the line, though careful positioning of the cursor will be necessary. While that symbol is displayed, press and hold ML, drag the edge to a new location, and release ML.
- Move the cursor to one of the corners (but not the upper right corner), positioning it so that it changes to a right angle into which an arrow points, though careful positioning of the cursor will be necessary. While that symbol is displayed, press and hold ML, drag the corner to a new location, and release ML. This approach to re-sizing of the window will automatically preserve the aspect ratio of the window.

Leave the window fairly large and overlapping the *Shell* window that is also open on the desktop.

Bring the *Shell* window to the fore by clicking ML somewhere in the visible portion of the window. Type the ⟨RETURN⟩ key a couple of times and note that there is no response. This window is still tied up in the running of the `gedit` command and will not be available for other uses until the *gedit* window is closed.

Close the *gedit* window.

15. *Prompt* 15% `history`

    A small number—by default 25—of the most recently executed statements is stored in a buffer. This statement displays the contents of that buffer. Each statement is labeled with a number—though the numbers are not necessarily consecutive—and the time of execution.

16. *Prompt* 16% ↑

    Typing the up-arrow key one or more times will retrieve a previous statement and allow you to edit the statement (by moving the cursor into the statement with the left (←) and right (→) arrow keys, by typing characters to be inserted, and/or by deleting characters with the backspace key). When you have the new statement properly constructed, typing the ⟨RETURN⟩ key will execute the statement. (Note that the cursor need *not* be at the end of the line when you type the ⟨RETURN⟩ key.) Retrieve a previous statement (say `ls`) and re-execute it.

17. *Prompt* 17% !*n*

    This statement—an exclamation point (often read "bang") followed by an integer—will re-execute the statement numbered n in the listing produced by the `history` command, though it offers no opportunity to edit the statement prior to execution.

18. *Prompt* 18% `gcalctool &`

    The operating system includes numerous desktop tools. This statement launches a calculator that, because of the ampersand that terminates the statement, is *detached* from the launching *Shell* window. The calculator has several sophisticated features and a variety of different modes—Basic, Advanced, Financial, Programming—accessible from selections in the Mode menu. To close the calculator, click ML on the '×' button in the upper right corner.

    Other useful desktop tools include `cal` (calendar), e.g., `cal 4 1938`, which will print the calendar for April, 1938, in the `Shell` window.

19. *Prompt* 19% `man ls`

    This statement displays the online `man`-page for the command `ls`. As with the command `more`, the space bar advances the display to the next screen; '`q`'—or (here for some reason) perhaps two of them—aborts the display.[23] Note the multitude of options available with the `ls` command and, in particular, note the role of the options used in the specific statement `ls -al`.

    Use this command to examine the man pages for several other UNIX commands, e.g., `man`, `cp`, `rm`, `mv`, `cat`, `more`, `head`, `tail`, `pwd`, `yppasswd`, `mkdir`, `rmdir`, `cd`, `gfortran`, `gcc`, `telnet`, `ftp`, `ssh`, `chmod`, `chgrp`, `find`, `grep`, and any others that catch your fancy.

20. *Prompt* 20% `mkdir test; cd test; pwd`

    This three-part statement (1) makes a new directory named `test` as a subdirectory in the default directory, (2) then—individual statements on a single line are separated by a semicolon—changes the default directory to that new directory, and (3) reports the current default directory to verify that the actions have been properly taken.

    Alternatively, the character '`\`' (followed by ⟨RETURN⟩) can be used to continue a long statement onto additional lines. In the present context, it makes little sense to enter the three statements above in the form

    ```
    mkdir test; \
    cd text; \
    pwd
    ```

    but the pattern at least illustrates the use of '`\`' to continue a statement (or statements) on a new line. The operating system prompts for each new line with a question mark '`?`'.

21. *Prompt* 21% `cd ..; rmdir test`

    This two-part statement restores the default directory to the original one and then removes the directory created at Step (20) above.

22. *Prompt* 22% `df` or `df -k`

    This simple statement produces a report on disk usage on the node at which you are working. All sizes are expressed in kilobytes.[24] The sample output

    ```
    Filesystem     1K-blocks     Used Available Use% Mounted on
    /dev/sda3       51606140 13993452  34991248  29% /
    /dev/sda2         495844   101953    368291  22% /boot
    /dev/sda1         204580      260    204320   1% /boot/efi
    NEWTON:/apps/   48660272 43599000   2549608  95% /apps
    /dev/sda5      206428032 58651776 137290240  30% /home
    newton:/home/  206428032 58651776 137290240  30% /home
    ```

    is a portion of the response to this statement on BRAHET. The file systems `/dev/...` are on the local system disk. All other disks reside physically on the server (NEWTON). Since a 1K-block contains 1024 bytes, the 206428032-block size of the disk on which the file system `NEWTON:/home` containing *your* files resides, for example, has a size of 206428032 kilobytes, i.e. 206 GB. At the time of the above printout, that disk was 30% full. Disk usage shuffles around from time to time, however, so the numbers presented in the report you generate are likely to differ from these.

[23] You can scroll up and down in the portion of the `man`-page already displayed by using the "page-up" and "page-down" keys.

[24] On the HP workstations, the kilobyte is the default unit; the option `-k` is present for compatibility with other versions of the UNIX operating system.

23. *Prompt* 23% `du` or `du -k`; `du -s` or `du -sk`

    These statements generate reports on disk usage at and below the current default directory. With the first statement in each pair, a line is printed for *each* directory. The line

    ```
    604     ./.gconf
    ```

    for example, informs the user that the directory `.gconf` in the current default directory uses 604 1024-byte (1K) blocks. With the second statement in each pair, the report contains only one line informing the user of the total use by the default directory and all directories below it. The version `du -s *` or `du -sk *` will generate a list containing the sizes of each subdirectory in the default directory. In some operating systems with some options, `du` reports sizes not only of directories and subdirectories but also of some individual files. Full details can be found in the `du` `man`-pages.

    You should adopt the habit of using these statements regularly to monitor your disk usage. Several ways to find and delete no longer needed files are described in Section 17.

24. *Prompt* 24% `find /apps/CPSUP -name "*lap*.f" -print`

    This statement will search the directory `/apps/CPSUP` and all subdirectories thereof for files whose name contains `lap` anywhere and ends with the file type `.f`, ultimately printing a list of any such files found.

25. *Prompt* 25% `???`

    Examine the effect of any other statements that catch your fancy.

While there is a *Shell* window on the desktop, we illustrate one more feature of the overall environment. To copy text from one place to another,

26. Type the statement `gunzip` and the ⟨RETURN⟩ key at the shell prompt. This action will generate an error message, i.e., it will display some well defined text on the screen for use in the next items.

27. Click ML *twice* on the word `compressed` in that error message. The word will be highlighted.

28. Click MM. The word will be copied to the current focal point for input. If the cursor is still in the *Shell* window, that point is at the prompt for the next statement. (Never mind that the word does not start a legitimate statement.)

29. Click ML *three* times on the line containing the word `help`. The line [including ⟨EOL⟩—the end-of-line character, conveyed by the extension of the highlighting all the way to the right edge of the window] will be highlighted.

30. Click MM. The line will be copied to the current focal point for input (right after the word `compressed` in the last line in the window). Since ⟨EOL⟩ was included, this operation will terminate a senseless statement, which will be processed by the operating system and generate an error message. Ignore that message.

31. Move the cursor to the start of the phrase 'Command not found.' in the second error message, press and hold ML, move the cursor to the end of the phrase, and release ML. The phrase will be highlighted, but—if you are careful in the selection—the highlighting will not extend across the screen and the character ⟨EOL⟩ will therefore *not* be included.

32. Click MM. The line will be copied to the current focal point for input. If ⟨EOL⟩ was included, this operation will terminate a senseless statement, which will be processed by the operating system and generate another error message, which you may ignore. If ⟨EOL⟩ was *not* included, clear the pending statement by typing ⟨CONTROL-U⟩ or by typing ⟨RETURN⟩ and ignoring the error message.

You have learned that *double* clicking ML on a word highlights the word, that *triple* clicking ML on a line highlights a line (including the end-of-line character), and that dragging ML across a segment of text highlights that segment. Further, you have learned that clicking MM with the cursor in a particular window—it need not be the *same* window containing the text—copies the highlighted text to the current insertion point in that window.

2.3 The Desktop

In the previous section, we illustrated some of the ways in which statements can be explicitly typed at an operating-system prompt in a *Shell* window. Users also have available to them all of the features of the *Desktop*, which occupies the entire screen on the workstation. This desktop is a stage for an icon-based, graphical user interface (GUI) to the operating system. The default desktop, called the GNOME desktop relies on window- and icon-based interfaces with the computer and offers the user a multitude of mouse-oriented ways to control the action of the computer.

1. Begin by *minimizing* all open windows by clicking ML on the 'Minimize' button labeled with the character '−' and located in the panel of buttons in the extreme upper right corner of each window.

   Notice that, when you minimize a window, it appears in the bottom panel as a small rectangle containing an icon identifying the application associated with the box and a brief textual phrase indicating the name of the application and/or the specific file (if any) that is being run on that application. A closed window can be reopened by clicking ML on the corresponding button in the bottom panel.

2. Now, double-click ML on the desktop icon labeled 'Computer',[25] thereby opening the *Computer* window. When opened, this window will display all accessible physical resources. In particular, the new window will contain icons labeled '500 GB Hard Disk' (the internal hard drive on your node), 'CD/DVD Drive', 'Filesystem', and perhaps others. We here explore only the file systems.[26]

   - Double-click ML on the icon in the *Computer* window labeled 'Filesystem' to display all of the folders that are currently available on the local node. The contents of some of these folders reside physically on the local node. The contents of others (e.g., `home` and `apps`) resides on the server. A few of the folders (lost+found, root, are marked with a light gray square embracing a black ×; these folders are privileged folders and are not accessible to non-privileged users.
   - Double-click ML on the `apps` icon and then on the `CPSUP` icon to open the directory `/apps/CPSUP`.[27]
   - Open and note the contents of any folder that catches your interest, paying particular attention to `/apps` (where the head directories for many third-party applications are stored), to `/apps/CPSUP` (where files associated with *CPSUP* are stored), and to `/etc` (where command files and macros associated with boot up and with setting the default configuration of user interfaces are kept).

3. Close each open window by clicking ML on the button labeled '×' and located in the extreme upper right corner of each window.

[25] In these contexts, we shall subsequently use the phrase "double-click ML on '…'" to mean "double-click ML on the icon labeled '…'".

[26] If there is no medium located in the CD/DVD drive, opening its folder will prove fruitless.

[27] The notation `/apps/CPSUP` conveys a *path* to the final directory. Specifically, `/apps/CPSUP` is a folder named `CPSUP` in the (higher-level) directory named `apps` which is itself a folder in the root of the entire file system for our network, which root is named simply `/`.

4. Double-click ML on the desktop icon labeled "home" to open it. This action brings up a window that gives you GUI access to the files and folders in your home directory. For simplicity in later discussions, we shall refer to this window containing icons for the files and folders in a particular directory as the *DirView* window. Even at your first login, there may well be a fair number of icons in this window. In particular, you should note icons for the `Desktop` and for the files `laplace.f`, `CPSUP.txt`, and `comments.txt`.[28] To become acquainted with the usage of the GUI to modify folders and files,

   (a) Double-click ML on `CPSUP.txt` to open it with the default program (`gedit`) associated with the file type `.txt`. You should now see the contents of the opened file in the *gedit* window. The name of the file is displayed on the tab at the top of the text and also in the header of the window. Close the window.

   (b) Use ML to drag the `CPSUP.txt` icon onto the Desktop, thereby granting quick and easy access to it.[29] Double-click ML on `CPSUP.txt` to open it, but close the window again.

   (c) Click MR on `CPSUP.txt` to bring up a list of options for that file. Select "Copy" from the list. Now, with the cursor in your `/home/`*YourUserName* folder, select 'Paste' from the EDIT menu. Your file should now be on the desktop and also in your home directory.

   (d) Press and hold ML on `CPSUP.txt` on the desktop and drag the icon to the 'Trash' icon (also on the desktop). Release ML on the 'Trash' icon to remove `CPSUP.txt` from the desktop. Note that the file isn't actually gone; it is merely placed in an area of the hard drive where it can still be accessed. Double-click ML on the 'Trash' icon and drag `CPSUP.txt` back onto the desktop. Open it to verify that it is unchanged. This method of deletion is relatively error-proof— if you accidentally move to the trash bin something that you really wanted to keep, you can easily restore it.[30]

   (e) Remove the file completely by clicking MR on `CPSUP.txt`, selecting 'Move to Trash' from the resulting pop-up menu, clicking MR on Trash, and selecting 'Empty Trash'.[31] The file has now been deleted completely and irretrievably.

   (f) Click ML on the 'Search' button in the upper right of the *DirView* window. This action gives access to a utility that is useful when you can remember the name of the file you are looking for but can't remember where it is stored. Unfortunately, the returned report does not very clearly indicate the path to the file that it finds.

5. The left-most menu in the top panel is the APPLICATIONS menu, which contains links to many useful programs installed on your workstation. Among them are

   - The ACCESSORIES submenu, which contains a variety of non-essential, miscellaneous applications, including
     - The calculator, a program which—through selections from the VIEW menu—allows a very wide range of mathematical, financial, and scientific calculations to be carried out. This program is set apart from other calculators (and is therefore worthy of note) by its ability to store user-defined functions, constants, and memory variables and recall them later.

[28] The window may also show a number of files and directories whose names begin with a dot—the so-called *hidden* files. Since these files are rarely of any interest, you may wish to suppress their appearance by making sure the item 'Show Hidden Files' in the VIEW menu of the *DirView* window is *unchecked*.

[29] Note that this operation actually *moves* the file to the desktop and *deletes* it from the original folder, so the item on the desktop is the *only* copy of the item in your directory structure. To *copy* the file to the desktop, click MR on the icon, select 'Copy', click MR on the desktop, and select 'Paste'. To create a link (UNIX terminology for what Windows calls a shortcut) on the desktop, click MR on the icon, select 'Make link', drag the resulting link—which will be placed in the same directory as the original—to the desktop, and (if you wish) click ML on the link, select 'Rename', and edit the name as you choose. Placing frequently used files or folders on the desktop can be especially convenient.

[30] This desktop protocol provides a level of protection against accidental and unintended deletions. In contrast, the `rm` command in the *Shell* window removes the file instantly, permanently, and irretrievably from your disk storage.

[31] Normally, GNOME will warn you that emptying the trash will delete all contents permanently. To proceed, simply press the 'Empty Trash' button.

- The text editors `emacs` and `gedit`. See Section 4 for more information on using these editors.
- The 'File Browser', which brings up the same tool that appears when ML is double-clicked on the 'home' icon on the desktop.
- The item 'Help', which brings up a screen in which you can browse for help in working with the GNOME desktop.
- The utility 'Screenshot', which is described in Section 7.
- A utility that provides capacity to search for files in your directory structure. See item 7 below.

- The GRAPHICS submenu, whose contents includes
  - The EVINCE document viewer.
  - The GNU GV (Ghostview) PostScript/PDF viewer, which can alternatively be invoked from a *Shell* window either with the command `gv` or the command `ghostview`. Described in Section 6.1.2, GhostView allows you to view PostScript files onscreen before printing them.
  - The GNU Image Manipulating Program `gimp`, which is a versatile photo- and image-editing program that is based on the same concepts as Adobe Photoshop. Although it can seem somewhat awkward at times, `The GIMP` is nonetheless a standard tool in the Linux world.[32]
  - `Tgif`, a program for creating two-dimensional drawings. The program is described more fully in Section 8.5.
  - The XV Image Viewer/Editor, which can crop and otherwise edit images but also can grab portions of the desktop for storage in files. The program is described more fully in Section 6.1.1.
- The most important items in the INTERNET menu are the web browser Mozilla Firefox, the item labeled 'FTP Client' (which launches the standard command-line ftp program in a *Shell* window), and the 'gFTP GUI Interface' (which launches an X-window interface to the ftp program).
- The OFFICE submenu, which contains
  - Adobe Reader 9, a program for viewing PDF files.
  - A dictionary.
  - The EVINCE document viewer
  - Several components of LibreOffice, a Linux emulation which provides something approximating Microsoft Office in a Linux environment. See Section 4.4 for more information on using this suite of programs.
  - The Xdvi previewer, which is described in Section 8.1.
- The SYSTEM TOOLS submenu contains at least one entry of relevance to the standard user, specifically the entry labeled 'Terminal'.[33] Since the terminal will almost certainly be the most used application, it would be prudent to create a shortcut in the top panel across the top of the desktop. If you have not already done so, create this shortcut by pressing and holding ML on the item 'Terminal' in the SYSTEM TOOLS menu, dragging the icon to a convenient spot in the top panel, and releasing ML. Once you have positioned this icon in that panel, clicking ML on it will open a new *Shell* window on the desktop.

6. Select 'Help' from the APPLICATIONS → ACCESSORIES menu to bring up a new window containing various help topics specific to GNOME, and spend some time familiarizing yourself with the contents of this manual. You might even wish to drag a 'Help' icon to the top panel for convenient access in the future.

[32] *Many* sources, including not only books but also web pages, provide information about and guidance on using The GIMP.

[33] Most of the remaining items in this menu are for advanced users and system administrators only; *changes made in some of these programs could render your account unusable.*

7. The PLACES menu next to the APPLICATIONS menu at the left end of the top panel contains shortcuts to locations with which you are already familiar: your *Home* folder, the Desktop, the Download directory, and Computer. It also contains a link to a search function, which itself utilizes a combination of the `find`, `locate` and `grep` commands.[34] This function will allow you quickly to find files anywhere on the computer. To use it, click ML on the menu item labeled "Search for Files..." and enter your desired search parameters (such as the file name `laplace.f`) and the *root* directory at which to begin the search. Starting at the level you specify, this program will search for the specified file, working its way through every folder "beneath" (inside of) the specified starting point. Clicking ML on the '+' sign in front of the text 'Select more options' will allow a refinement of the search.

8. At the right end of the top panel are at least two icons and a menu titled with your name (as it appears in the comment field in your line in the password file). Note that

   - CLicking ML on the speaker icon brings up a slider in which you can adjust the volume while clicking MR on that same icon brings up a menu in which, among other things, you can mute or unmute the speaker.
   - Clicking ML on the icon that looks like two overlapping monitor screens brings up a menu of options for wired connections while clicking MR on that same icon brings up a menu of options associated with networking. *Changing any of the default options in these menus could render your session non-functional.*
   - Clicking ML on your name brings up a menu in which, among others, you will see an item named 'Log Out ...'. To facilitate logging out later, you may wish to
     - Click ML on the *YourName* menu, and
     - Drag a 'Log Out' icon to a convenient spot in the top panel.

     Fuller detail on the process of logging out is presented in Section 2.4 for more detail.

9. Click ML on the item 'System Settings' in the menu accessed by clicking ML on your name at the far right end of the top panel (or, alternatively, select 'System Settings' from the APPLICATIONS ⟶ SYSTEM TOOLS menu. The resulting screen provides access to a number of utilities for configuring features of your system. For example, clicking ML on[35]

   - The icon labeled 'Background' provides the resources to set the background display for your sessions. The drop-down menu above the panel of possibilities offers three options, 'Wallpapers', 'Pictures Folder', and 'Colors & Gradients'. You can select a background from the available wallpapers or select one from your 'Pictures' folder. Alternatively, clicking ML on the '+' sign near the lower left corner brings up a browser in which you can locate a desired image. Images may, of course, be downloaded from your camera or from network and Internet sites and stored in your 'Pictures' folder to be used as wallpaper.
   - The icon labeled 'Brightness and Lock' allows you to set the time interval of inactivity after which the screen will go dark. This behavior apparently substitutes for the invocation of a screen saver. (The option to lock your screen offered here has been disabled system wide.)
   - The icon labeled 'Displays' provides for setting the resolution on the screen, the default being '1680×1050 (16:10)' resolution and 'Normal' rotation. Change these options at your peril. Some of the options in this screen have been disabled.
   - The icon labeled 'Keyboard' brings up a screen with two tabs. In the 'Typing' tab, you customize several aspects of the behavior of the keyboard and the on-screen cursor; in the

[34] These commands are very useful when passed directly to the shell. Read the man-pages on them for more information.

[35] To return to the *System Settings* window from the screen associated with any icon in that window, click ML on the icon/button marked with a 3×3 array of black square located in the upper left cornet of the open window.

'Shortcuts' tab, you can customize the definitions of a mountain of keyboard shortcuts. Feel free to tweak these settings as you see fit, as efficiency can be gained by being able to perform actions rapidly without using the mouse. When you're done modifying these shortcuts, close the *Keyboard Shortcuts* window.

- The icon labeled 'Mouse and Touchpad' allows you to select whether the mouse will be right- or left-handed, set the relationship between mouse motion and cursor motion, specify how long the system will wait after you press and hold a key before repeating the key, and set how rapidly the key will repeat when held.
- The icon labeled 'Sound' provides for setting the output volume, muting the sound, setting the character of warning sounds, and other features.
- The icon labeled 'Universal' provides tools to conform the behavior of the system to particular disabilities of the user.

Now that you have completed the orientation to the Linux desktop, you are urged to clean up your desktop and arrange it in a sensible manner (i.e. remove any icons that aren't being used, re-position the remaining icons, etc.). Keep in mind that much of the top panel is available to contain shortcuts to frequently-used applications. Placing shortcuts here will save you time later, as you won't find yourself searching through menus and directory trees trying to find the program you want.

2.4 Logging Out

Before logging out, please routinely check your disk usage and remove any files that you no longer need to keep. The first time you reach this point in this document, you probably should execute the statement

```
rm comments.txt laplace.f CPSUP.txt
```

to invoke the command `rm` (remove) to delete the sample files created during your first session. If you created other sample files, please remove those as well, either by executing the statement `rm` again with additional file names or by adding those file names to the list in the above statement. By default at Lawrence, the statement `rm` asks for confirmation before deleting each file in the list, though that default can be overridden *if you are confident you have provided a correct list of file names.* [36] In subsequent sessions, cleaning house at the end of the session may involve more extensive deletions. Ways in which those deletions can be achieved more efficiently are described in one of the items in Section 17, "Rules of Citizenship within the CPL".

You may also wish to implement one additional customization. By default, the configuration of your account sets parameters so that the state of your session[37] when you log out will be saved and restored when you next log in. Unfortunately, *that behavior is not 100% reliable, especially if your session has several windows open.* To turn off that feature,

- Start the program `gnome-session-properties` by typing the statement

  ```
  gnome-session-properties &
  ```

- In the tab labeled 'Options', make sure the box labeled 'Automatically remember running applications when logging out' is *un*checked.
- Click ML on 'Close'.

[36] The statement `rm -f` *filename(s)* will suppress the requests for a confirmation of each deletion.

[37] Your session embraces the programs you have running at any time but does not include anything in the top and bottom panels or the 'Computer', 'home', and 'Trash' icons on your desktop.

When you next log in, your desktop will be completely empty except for the top and bottom panels and the three icons identified above.

If, in addition, you then wish to start up a specific program when you create a new session, you should

- Start the program `gnome-session-properties` again.
- Select the 'Startup Programs' tab.
- Click ML on the 'Add' button.
- In the pop-up box titled 'Add Startup Program',
  - Enter a mnemonic name—e.g., 'Shell Window' if you wish to start a *Shell* window as your session is launched—for the desired program in the box labeled 'Name'
  - Enter the statement—e.g.,

    ```
    /usr/bin/gnome-terminal --geometry=80x24+25+500
    ```

    that should be executed[38]—in the box labeled 'Command'.
  - Enter a comment—e.g., 'Starts Shell Window'—in the box labeled 'Comment'.
  - Click ML on the button labeled 'Save'. The new program will be added to the list in the 'Startup Programs' tab and the command will be executed when a new session is created.
  - Click ML on 'Close'.

Once you have deleted any files you no longer need to keep and, if you choose, have effected the adjustments described in the previous two paragraphs, you are ready to log out as described in the next paragraph. Be aware that merely *closing* windows does *not* effect logout. Be aware also that walking away from a workstation *without logging out* and *without removing your possessions from the area* leaves the workstation and the area in a state that is awkward for the next user and leaves your directory structure vulnerable to inadvertent—or, if you are unfortunate, perhaps even malicious—corruption. *The rules of citizenship in the CPL—see Section 17—expect that you will logout, remove your possessions, and return all manuals you have used to the bookshelves in the CPL whenever you leave a workstation.*

Once you have completed your session and cleaned up your account and your desktop, you are ready to log out. To do so,

- Open the menu headed by your name and click ML on the item 'Log Out ...' or, if you created a 'Log Out' icon in the top panel, clicking ML on that icon.
- Click ML on 'Log Out' in the confirmation window that comes up.

Presently, the login screen for the next user will appear on the monitor.

3 The User Environment

The environment presented to the user in a *Shell* window depends in the first instance on the particular *shell* that is used. Many shells are in common use, and a few are available on the machines in the CPL. At Lawrence, the *default* shell is the *t*-shell, which is an elaboration of the traditional

[38] Here, the window created will be 80 characters wide by 24 lines high and will be positioned with an X-offset of 25 and a Y-offset of 500. The units of the third and fourth numbers are not clear, but the values here illustrated place the *Shell* window in the lower left corner of the desktop.

c-shell.[39] The details of the user environment in the *t*-shell are determined by three ASCII files,[40] each of which contains UNIX statements that influence the behavior of the shell. These files are *sourced*—the UNIX verb for execution of a command file or script—in the order[41]

- `/etc/csh.cshrc`
- `/etc/csh.login`, which includes sourcing the locally added file `/etc/profile.d/lu.csh`,
- `$home/.cshrc`

whenever a new shell is established. Among other things, sourcing of the first of these files sets the *mask* so that all newly created files are by default given fairly restrictive permissions (`rwx` for the user; no access for others); sets the form of the operating system prompt; sets the history variable so that the previous 25 statements will be remembered (and can be retrieved by pushing the up-arrow key at the system prompt); displays the message of the day (if any); establishes the default value for the `PATH` environment variable to include directories necessary for the running of standard installed software; and, finally, defines several convenient aliases, some of which are listed in Table 1.

In its default form, the third of these files (`.cshrc` in your home directory) is simply a template containing no executable statements. This file can, however, be edited by each individual user to incorporate additional customization of the shell environment. For example, if you add below the row of number signs the lines

```
cd ~
rm -f .history
set prompt='%m %h% '
```

you will make sure that each newly created *Shell* window will be launched with your home directory as the default (first line), the numbering of prompts in each *Shell* window will start at 1 (second statement), and the prompt at each command in each window will be composed of the node name and the number of the command since that window was opened (third statement).

Further information about statements that can be submitted to the shell can be found in the `tcsh man`-page, in Kernighan and Pike (Chapters 3 and 5), and in Sobell (Chapters 10 and 11).

4 Text Editing

Several text editors are available on the workstations in the CPL.

4.1 Gedit

The mouse-oriented, GNOME-specific text editor Gedit is, in most contexts, the default editor in the Linux environment. Among other features, Gedit includes syntax highlighting and other useful features, such as the ability to accept standard input through "piping". To orient you to this editor, we present here a quick session—with comments.

[39] Other shells include the Bourne Again Shell (`bash`) and the *z*-shell (`zsh`). Details on changing shells are beyond the scope of this guide.

[40] Because they are ASCII files, these files can be displayed on the screen using the `cat` or the `more` command—and you may wish to examine them so you have a sense of how they are structured.

[41] The specification `$home` in some of these file names refers to the user's home directory `/home/`*YourUserName*. Because the character string `home` is defined as an *environment variable* for your home directory, the string `$home` can itself be used in statements submitted to the operating system.

Table 1: Some of the default aliases in the physics network at Lawrence. Others will be defined at appropriate points later in this guide.

'ls'	for	'ls -C'	Displays file names *in columns*
'rm'	for	'rm -i'	Modifies rm to request confirmation
'cp'	for	'cp -i'	Inquire before overwriting
'mv'	for	'mv -i'	Inquire before overwriting
'lo'	for	'logout'	Short-hand for logout
'ghostview'	for	'gv'	Launch PostScript viewer gv
'lpdup'	for	'lp -dCPL_HP4200dup'	Directs print job to HP monochrome printer for double-sided printing
'lpcol'	for	'lp -dCPL_HP4700'	Directs print job to HP color printer for single-sided printing
'lpcoldup'	for	'lp -dCPL_HP4700dup'	Directs print job to HP color printer for double-sided printing
'ssh -X'	for	'ssh'	Launch ssh so X windows can be displayed on the local machine by the remote host
'TOE'			Displays Theory of Experiment
'CPSUP-PYMTM'			Displays Python/Mathematica version of CPSUP
'CPSUP-IDLMPL'			Displays IDL/MAPLE version of CPSUP
'CPSUP-OCTMAX'			Displays OCTAVE/MAXIMA version of CPSUP
'LATEX'			Displays LATEX appendix from CPSUP
'NCM'			Displays Notes for Computational Mechanics
'LG'			Displays Local Guide

---

1. *Prompt* 75% `gedit newfile.txt &`

   This statement launches a *Gedit* window containing the contents of the specified file, here `newfile.txt` (which is, of course, initially non-existent). The file and, hence, the *Gedit* window contain no text.[42] To create a simple file,

   (a) Move the cursor into the (now open) *Gedit* window and type a few lines, making sure to conclude the last line by typing the ⟨RETURN⟩ key. Note, incidentally, that the cursor is by default a vertical line that is positioned *between* two characters in the file, not *on* a character.[43]

   Note that, by default, the line will wrap at the width of the window but no hard return will be inserted. A file created with this option may, when opened in some other text editors, consist of very long lines that run out of the window and off the screen.[44]

   (b) Save the file you have created by selecting 'Save' from the FILE menu or by typing ⟨CONTROL-S⟩.

   (c) *Without closing* `newfile.txt` or the *Gedit* window, move the cursor into the *Shell* window.

[42] Note that Gedit does not run detached from the launching window unless the statement starting the program is concluded with an ampersand (&).

[43] The key labeled 'Insert' in the area of the keyboard to the right of the main keyboard toggles the cursor between a vertical line and a block. When the cursor is a vertical line, typed characters are inserted into the text; when it is a block, each typed character overwrites the identified character in the text.

[44] This behavior can be changed by accessing the 'View' tab in the EDIT → PREFERENCES menu.

2. *Prompt* 76% `cat /apps/sysmgr/lu.csh | gedit &`

   This statement opens the file `lu.csh` (a login configuration file) in Gedit, but on a second *tab* in the *Gedit* window, not in a fresh *Gedit* window.[45,46,47] *For the rest of this exercise, keep the window displaying* `lu.csh` *as the active window in Gedit.*

   (a) Bring the *Shell* window to the fore and type the statement `more /etc/csh.cshrc` to display some text on the screen.

   (b) Select several lines of text from the display in the *Shell* window by moving the cursor to the beginning of the first line, pressing and holding ML, moving the cursor down the screen to highlight the desired text, and releasing ML.

   (c) Move the cursor back into the *Gedit* window (you need not bring it to the fore, though you may), and then click MM. The highlighted text in the *Shell* window will be copied into the *Gedit* window at the current position of the input focus.

   (d) By default, the menu bar in the *Gedit* window exhibits the File, Edit, View, Search, Tools, Documents and Help menus. Look at all these menus (including the Help menu), and explore any capabilities that catch your fancy. Note, in particular,

   - The item 'Highlight current line' in the 'View' tab of the Edit → Preferences menu. Indeed, you may want to make sure the box in front of that item is checked so that it will be easy for you to locate the line in which the cursor resides.
   - The submenu Highlight Mode in the View menu. This item offers numerous options for highlighting the text in different ways. Note, for example, that by selecting the subsubmenu Sources, you can highlight all IDL or C or Fortran 95 commands in the currently displayed text or, by selecting the subsubmenu Markup, you can highlight all LaTeX commands. To illustrate this feature, make sure the `lu.csh` file is the active file and select 'sh' from the subsubmenu Scripts. You should notice that its display has acquired some color coding. All of the syntax—never mind if it is not currently meaningful to you—is now highlighted specifically for shell scripts. This feature is particularly useful as an aid to catching typographical and syntactical errors. Note, incidentally, that in some cases Gedit will select an appropriate highlighting by noticing the file type of the open file.
   - The self-explanatory (and quite useful) item 'check spelling' from the Tools menu.
   - The item 'Go to Line' in the Search menu, which allows you to specify a line to which the cursor is to be moved. If, for example, you have an error on line 56, you simply select 'Go to Line' from the Search menu and type 56 in the box that pops up. The line is immediately identified without any further action on your part, and you will be able to edit the identified line. The 'Go to Line' window remains open until you make an edit. If you wish to search for an additional line, you will then have to reopen 'Go to Line', either by re-accessing the menu or, more conveniently, by typing ⟨CONTROL-I⟩.
   - The item 'new' in the File menu. Selecting this item will open a new, blank document in a new *tab*; it will *not* launch another instance of Gedit.

   (e) When you are finished, close the *Gedit* window by selecting 'Quit' from the File menu. You may have to indicate explicitly that you don't want to save any further files before the program will actually terminate.

3. *Prompt* 77% `ls -al`

   The output will display all of the files in your home directory and their permissions, along with the default permission for the newly created file, `newfile.txt`.

[45] Because the file is opened in an existing *Gedit* window that is already detached from the launching *Shell* window, the ampersand at the end of this most recent statement is unnecessary, though it does no harm.
[46] You can switch from one file to the other by selecting the corresponding tab.
[47] Because you do not have write access to the file `lu.csh`, it is displayed in Gedit as a new and as yet unsaved file.

4. *Prompt* 78% `rm newfile.txt`

   This statement will remove the (presumably junk) file created in the above session, though you will have to respond '`y`' to the request for confirmation before the task will be accomplished.

To edit an existing text file, `gedit` can also be invoked by double clicking ML on the icon for that file in the *DirView* window on the desktop. Further, if `gedit` happens to be open, you can open an existing text file from within `gedit` by selecting 'Open...' in the FILE menu. Alternatively, if you have moved the 'gedit' icon onto your desktop, `gedit` can be launched by double clicking on that icon or by dragging the icon representing the file from the *DirView* window to the 'gedit' icon. Finally, `gedit` can be launched by selecting it from the APPLICATIONS → ACCESSORIES menu in the top panel.

Additional information about `gedit` can be found in the HELP menu *within* `gedit` and in the `gedit man` page.

4.2 Emacs

The editor `emacs` belongs to the GNU family of programs, all of which are publicly available under the terms of the GNU general public license and are widely used on many different platforms. This program is launched with a statement of the form[48]

`emacs` *filename*

This program is extremely customizable, allowing the user to set colors, fonts, font size, window size, etc. It supports the editing of multiple files (several buffers simultaneously) and allows split screens and can force files automatically to conform to standard formats, for example, for C or FORTRAN programs. It has two word-wrapping options available, one of which inserts carriage returns between words while imposing a user-specified maximum line length and the second of which simply wraps the text at the right side of the window without adding carriage returns. The 'search and replace' function asks by default for confirmation at each instance of the search text.

Online help for `emacs` can be accessed from the HELP menu in the tool bar of the program window. In addition, the `man`-page for this program describes the program's command line options, and many books on UNIX (e.g., Sobell, Chapter 9) contain further information.

One particularly useful feature of this program is its ability easily to reformat a text file containing excessively long or unacceptably short lines so that all lines have "reasonable" length. To achieve that reformatting,

- Open the file in the program.
- Highlight the portion of the text you wish to reformat.
- Type ⟨`ALT/q`⟩.
- Save the reformatted file by selecting 'Save *filename*' from the FILE menu.
- Exit from the program by selecting 'Quit' from the FILE menu.

4.3 Vim

The standard UNIX editor `vim`, (an improved version of the now out-dated `vi`) is invoked by the statement

[48] The filename is optional. If launched with no filename, the program starts with an informational display describing several useful features.

`vim` *filename*

With the new improvements made to `vim`, it has become a widely used text editor. Its use is described in the `vim man`-page and in many books on UNIX and Linux operating systems. In case you happen to start the program, we mention here only one item: To exit from the program, type ':q' or, if you happen to have changed the file you opened, ':wq' to save the changes or ':q!' to exit without saving the changes.

4.4 LibreOffice

The LibreOffice suite of programs containing a Word-like editor, a spreadsheet, and other components has been installed on the workstations in the CPL. All components are accessible from a control window that is launched from a *Shell* window with the command

```
soffice &
```

which is aliased to the proper executable, or by selecting 'LibreOffice xxxx' from the Applications → Office menu in the top panel. Once you have started the manager for the LibreOffice suite with `soffice`, you can open a new file by selecting the appropriate file type from the File → New menu. On-line help messages will be useful as you start to work with this program. The intent, however, is for this suite to mimic the behavior of Word, Excel, PowerPoint, and other components of the Microsoft Office suite—and for the files produced with the Microsoft programs to be compatible with those produced with LibreOffice.

5 Printing Files

The CPL is equipped with a monochrome laser printer (Hewlett Packard P4015x)[49] and a color laser printer (Hewlett Packard 4700dn). All printers understand PostScript and ASCII text files and can print on paper and on *the proper* transparency materials.[50] Color copies cost rather more than black and white copies, so users are requested to make only those color copies that are necessary. Examine candidates on the screen before selecting the few to be reproduced in hard copy.

A word regarding double-sided printing is in order. Sheets already printed on one side with either Hewlett Packard printer can safely be passed through either printer for printing on the other side. One can, for example, print monochrome on one side and then color on the other side. A bit of experimenting may be necessary to determine how to insert the sheet properly in the second printer.

5.1 Printing ASCII text files and Monochrome PostScript Files

The HP P4015x laser printer is defined as a network printer (i.e., it has its own IP address) on the physics network. Because this printer for simplex printing is defined as the default printer, the standard printer command `lp` can be used to direct both ASCII text files and PostScript files to the printer. Indeed, since this monochrome printer has been defined as the default printer, the user need not even specify the destination. A simple statement like

`lp` *filename(s)*

[49] For historical reasons, the queue names for the monochrome printer retain the model HP4200 of its predecessor.

[50] The *wrong* material may well melt on the fusing roller and result in a major internal mess (and a major repair bill). *Please don't use the wrong material.*

is sufficient to direct a file (or files) to this printer.[51] An on-screen message—something like `CPL_HP4200-247`—will inform you of the ID for the submitted job.[52] Should you, after submitting a job, decide to cancel the job at any time before the file has been completely transmitted to the printer, you may type the statement

`cancel` *JobID*

at the shell prompt.

The statement described in the previous paragraph will print the file using only one side of each sheet of paper (*simplex* mode). The monochrome laser printer can also print automatically in *duplex* mode. To invoke that mode, you must submit the file to a logically different but physically identical printer. Since the "duplexing" printer is not the default, it must be explicitly specified in the statement that submits the job. In full, the statement will be

`lp -dCPL_HP4200dup` *filename(s)*

but the alias `lpdup` has been defined so that the simpler statement

`lpdup` *filename(s)*

will also work.

ASCII text files and monochrome postscript files can also be directed to the HP color printer, though their printing on that device will be somewhat slower than their printing on the monochrome printer. The necessary statements are described in Section 5.2, and the printer will recognize automatically that the file is to be printed in black and white.

The local shell scripts `prtex` and `prtexdup`, which are defined by files stored in the directory `/apps/local`, are also available. These scripts are invoked with the statements

`prtex` *filename* or `prtexdup` *filename*

and use the two files `document.start.tex` and `document.end.tex`, which are also stored in the directory `/apps/local`. Each takes a plain ASCII text file as input and processes it as if it were a LaTeX document in a `verbatim` environment.[53] The result is the same as printing the text file directly with `lp` or `lpdup`, but with a slightly smaller font and with margins better suited for insertion of the copy into a three-ring notebook. Note, in particular, that the script creates a few files named `temporary.*` in the default directory, though these files are removed by the time the script has completed execution. Note also that these scripts can*not* process a *list* of files; each file in the list must be submitted with a separate command.

5.2 Printing Color Files

Color PostScript files can be submitted to the Hewlett Packard color printer (HP4700), which is defined as a network printer with its own IP address on the physics network. The driver for this printer recognizes only straight ASCII text files and color PostScript files. All other formats (`.rgb` files produced, for example, by screen copying as described in Section 7) must be converted to PostScript with a program like `xv` (See Section 6.1.1) before being directed to this printer. The simplest way to direct ASCII or PostScript files to one of the color printers is to invoke one of the statements

[51] The fuller statement `lp -dCPL_HP4200` *filename(s)*, which includes the explicit specification of the default destination, is equivalent.

[52] The statement '`lpstat`' will, by default (no options) produce a report on the status of all your submitted print jobs. Though (curiously) this report doesn't include the filename, it does display the job ID and can be used to remind you of forgotten ID's.

[53] For more about LaTeX, see Appendix A in *CPSUP*.

`lp -dCPL_HP4700` *filename(s)*
`lp -dCPL_HP4700dup` *filename(s)*

for single-sided and double-sided (duplex) printing on the HP4700. To simplify this process, the aliases

@`lpcoldup` `lpcol` *filename(s)*
`lpcoldup` *filename(s)*

have been defined to provide access to the two queues.

Color files can also be submitted to the *monochrome* printer, but—understandably—the printer will render those images in a grey scale rather than in full color.

6 Viewing Files on the Screen

Text and images can be stored in files with a wide variety of formats. Utilities and other programs available on the HP workstations are capable of displaying many of these formats on the screen of the workstation. The simplest files, of course, are ASCII text files, which can be displayed in a *Shell* window with the `cat` and `more` commands and opened in a wide assortment of text editors (and—additionally—printed with the `lp`, `lpdup`, `lpcol`, `lpcoldup`, `prtex`, and `prtexdup` commands).

6.1 Tools Available

6.1.1 Xv

The program Xv[54] is described in its documentation as an "interactive image display program for the X-window system". Invoked with the statement[55]

`xv` *filename*

the program will display graphics files of most formats and will perform conversions among them. Supported file types include GIF (file type '`.gif`'), TIFF (file type '`.tiff`'), RGB (file type '`.rgb`'), JPEG (file type '`.jpg`'), X-11 Bitmap (file type '`.xbm`'), Windows Bitmap (file type '`.bmp`'), PPM (file type '`.ppm`'), and PostScript (file type '`.ps`'). Thus, `xv` can be used, for example, to convert other formats to PostScript so the image can then be printed or included in a LaTeX document. The program also has numerous graphics editing functions, ranging from simply changing the dimensions of an image to editing it pixel by pixel or changing the color map. The program has both a graphical interface and a spectrum of command line options. A detailed description of all features of `xv` is contained in an *extensive* printed manual, a copy of which is in the CPL; the `man`-page is merely a pointer to the printed document.

While you may encounter any of the supported file types, a few sample `.rgb`, `.gif`, and `.jpg` files for exploring `xv` can be found in several directories, including `/apps/sysmgr/images` and `/apps/CPSUP/lg/images`. For example,

[54] Copyright (1994) by John Bradley. The version at Lawrence is 3.10a which, except for a few patches, is the current version. The first time you open the *xv controls* window, you should click ML on the 'About XV' button and read the copyright statement, which permits unregistered personal use of the software for purposes of exploration and "playing" but requires registration if `xv` is used in the course of your work. Up-to-date information is available at the URL `www.trilon.com/xv`.

[55] The filename is optional (once you have the program running, you can open any files you want to edit). The filename may include wild card characters, in which case `xv` opens all matching files.

1. Start the program with the statement `xv` at a shell prompt and position the *xv* window in a convenient spot.
2. Place the cursor in the *xv* window and bring up the *xv controls* window by typing a question mark (?) or clicking ML.
3. Place the *xv controls* window in a suitable spot.
4. Click ML on the 'Load' button in the *xv controls* window to bring up the *xv load* window.
5. Enter the (full) name, e.g., `/apps/CPSUP/lg/images/fract004.gif`, of the file to be loaded.
6. Click ML on the 'OK' button. After a moment, the *xv load* window will disappear and the image will appear in the *xv image* window, replacing the Xv logo that was in that window initially.
7. (If you wish) Save the image in a chosen format by
   - Clicking ML on the 'Save' button in the *xv controls* window.
   - Select the desired output format in the resulting *xv save* window.
   - Enter the desired file name (with appropriate file type) in the 'Save file' dialog box.
   - Click ML on the 'OK' button in the *xv save* window. This action may bring up an additional window, which—if it appears—will give you an opportunity to specify several aspects of the saved image. Clicking ML on the 'OK' button in this window initiates the actual saving of the image in the specified file.
8. Click MR with the cursor in the *xv image* window to close the *xv controls* window. (Actually, MR toggles back and forth between opening and closing the *xv controls* window.)
9. To exit from the program altogether, click ML on the 'Quit' button in the *xv controls* window.

6.1.2 GhostView

The program GhostView is an X-window program which will display PostScript and, apparently, PDF files. While there are many command line options that influence the behavior of GhostView (see the `man`-page), the simplest way to display a PostScript file involves starting Ghostview with the statement[56]

`ghostview` *filename* or `gv` *filename*

which opens a window in which the file is displayed. Strictly, GhostView works together with the "real" display program GhostScript to create the view: GhostScript interprets the file; GhostView creates the display window and facilitates user interaction with GhostScript. Having GhostScript on the system allows other programs, like `xv` and `xdvi`, to generate views of PostScript files as well. Though programs make use of it, the user will probably never call GhostScript directly.[57]

Several PostScript files you can use to explore the capabilities of GhostView can be found with names like

`/apps/CPSUP/lg/images/fract0??.ps`	(?? = 05, 12, 19, 29)
`/apps/CPSUP/lg/plate?.ps`	(? = 1, 2, 3, 4)
`/apps/CPSUP/lg/ColorPlate?.ps`	(? = 1, 2, 3, 4)

As an exercise, you might want to use Xv as described in Section 6.1.1 to convert one of the fractal '`.gif`' files to PostScript for viewing with GhostView. Note that the resulting file is *big*, so please remove it (with `rm`) after you have examined it.

[56] Technically, the program is known as `gv`, but we have set `ghostview` as an alias for `gv`. To bring up the man page, however, you must type the statement `man gv`.

[57] GhostView and GhostScript are publicly available programs and can be installed and used without permission. Information is obtainable at the URL `www.cs.wisc.edu/~ghost`.

6.1.3 Acrobat

Adobe Acrobat[58] is a program that, in its free version, can read files in the PDF format (PDF files, usually with file type `.pdf`) and, in its fuller version, can both read and write PDF files. Adobe Acrobat Reader (the free version), which is installed in the CPL, will be launched by many applications when a file you have selected for download or display is a PDF file. Since this format is used by many online physics journals, you are particularly likely to confront Acrobat when you are examining various online technical publications. The PDF reader can also be launched directly from the shell prompt with the statement `acroread`. If a specific file name is not appended to the statement, the program will be launched but will require that you select a file from a browser.[59] For PDF files containing no internal hyperlinks,

- The window used by Acrobat Reader has two panels, though the left panel is so narrow as to be barely noticeable (and has very limited function). The text is contained in the *much* wider right panel.
- The window containing the text *may* by default be set so that the display jumps as it scrolls from page to page. An alternative setting, in which the scrolling runs smoothly and text straddling two pages can be more readily displayed, can be achieved by opening the EDIT → PREFERENCES menu, selecting the category 'Page Display' from the list of categories along the left edge of the resulting window, making sure that boxes labeled 'Page Layout' and 'Zoom' near the top of the window are both set to 'Automatic', and clicking ML on the button labeled 'OK' at the bottom of the window.

If, on the other hand, a more elaborate PDF file containing hyperlinks is opened with the Acrobat Reader, the left panel will be wider and will contain a hyperlinked table of contents for the document displayed in the right panel. Further, internal cross references in that document, including those in a table of contents and an index, may themselves be hot links to the referenced portions of the document. The width of the left panel can be controlled by dragging a (barely visible) handle at the (vertical) middle of the right edge of the panel.

Note that migrating among a succession of links already accessed can be achieved by using ⟨ALT- ⟵⟩ and ⟨ALT- ⟶⟩.

6.1.4 The GNU Image Manipulation Program

The GNU Image Manipulation Program (GIMP), an open-source image-editing program similar in design and function to Photoshop, is a standard application throughout the open-source world. This program allows the user to create nearly photo-realistic images, while at the same time providing the basic functionality that such programs as MSPaint offer. The GIMP is capable of

- converting images from one file format to another (e.g. JPEG's to GIF's) as well as converting image file formats to color PostScript files that can be sent to the local printers;
- precision editing of photographs, data tables, video frames, or any other image;
- creating and rendering very complex images from scratch with relatively little hassle.

This list is by no means all-inclusive. The `GIMP` can be started by selecting 'GNU Image Manipulation Program' from the GRAPHICS submenu of the APPLICATIONS menu or by typing the statement `gimp` at the prompt in a *Shell* window. More information on running The `GIMP` is available in countless books and internet sites, as well as the internal HELP pages.

[58] Available from Adobe Systems, Inc.

[59] The browser may be displayed automatically and by default, but can always be brought up by selecting 'Open' from the FILE menu.

6.1.5 Evince Document Viewer

The `Evince Document Viewer` is a Linux-specific program that is able to read many different formats of documents, including PDF and PostScript. When this program opens PDF files, however, it converts them to a file type unique to itself, '.bqy'. To access this program, select 'EVINCE Document Viewer' from the GRAPHICS submenu of the APPLICATIONS menu or type the statement `evince` at the prompt in a *Shell* window. Across the top of the *Document Viewer* window are

- A menu bar with items 'File', 'Edit', 'View', 'Go', 'Bookmarks', and 'Help', and
- A navigation bar containing tools to migrate in the document.

The bulk of the window comprises the area in which the document itself will be displayed. That area, in turn, will be divided into two vertical panels, the left one containing reduced images of each page in the document and the right one displaying more readable copies of the selected portion of the document. The content in each of these panels can be separately scrolled and clicking ML on a page in the left panel will instantly display that page in the right panel.

Details about the features and operation of the `Evince Document Viewer` can be found in the program's HELP menu.

6.1.6 Image Viewer

The Eye of GNOME image viewer can be opened by clicking ML on 'Image Viewer' in the APPLICATIONS → GRAPHICS menu. It offers the ability to read and display a wide number of image formats[60] and to save in a smaller number of formats.[61] Much more information can be obtained by accessing the help menu within the program.

7 Copying Portions of the Screen

A variety of programs exists to facilitate copying entire windows or portions of windows into files, typically bit-mapped files. To exploit the capability of Xv to capture all or part of a window, start the program by typing `xv` at a UNIX shell prompt, use ML to position the outline of the *xv image* window in an out-of-the-way place, and then

1. Make sure the portion of the screen you wish to grab is completely displayed on the screen and not covered by any other window.
2. Bring up the *xv controls* window by typing a question mark (?) or clicking MR with the cursor in the *xv image* window, and position it so that it does not overlap the window to be grabbed.
3. Click ML on the 'Grab' button in the *xv controls* window to bring up the *xv grab* window and make sure that it, too, does not overlap the window to be grabbed.
4. Accepting the 0 sec delay, click ML on the 'Grab' button in the *xv grab* window. The program will beep once immediately to signal that it is ready to "grab" a window, and the *xv grab* window will disappear from the screen.

[60] ANI - Animation, BMP - Windows Bitmap, GIF - Graphics Interchange Format, ICO - Windows Icon, JPEG - Joint Photographic Experts Group, PCX - PC Paintbrush, PNG - Portable Network Graphics, PNM - Portable Anymap from the PPM Toolkit, RAS - Sun Raster, SVG - Scalable Vector Graphics, TGA - Targa, TIFF - Tagged Image File Format, WBMP - Wireless Bitmap, XBM - X Bitmap, and XPM - X Pixmap.

[61] BMP - Windows Bitmap, ICO - Windows Icon, JPEG - Joint Photographic Experts Group, and PNG - Portable Network Graphics.

5. At this point, the user has two options:

   - To grab an entire window, click ML anywhere in the window.
   - To grab a portion of a window, move the cursor to the upper left corner of the desired area, depress and hold MM, move the cursor to the lower right corner of the desired area (note the rubber band box thus produced), and release MM.

   In either case, after a moment, the program will beep a second time and a copy of the grabbed window will appear in the *xv image* window, replacing the xv logo that was in that window initially.

Once grabbed, of course, the image can be saved in any of the supported formats by the procedure described in Section 6.1.1. The program Xv also permits cropping and other adjustments of the image.

To copy the *entire* screen in the Linux world, simply press the 'Print Scrn/SysRq' button located third from the right-hand end in the top row on the keyboard. To take a screen shot of just the *active* window,[62] hold ⟨ALT⟩ and press 'Print Scrn/SysRq'. To take a screen shot of a selected area, hold ⟨SHIFT⟩ and press 'Print Scrn/SysRq'. The resulting saved file will be given a lengthy name that includes the date and time of creation, will have file type '`.png`', and will be stored in the PICTURES subfolder of your home directory. You can, of course, rename it in that subfolder and/or move it to some other location.

Alternatively, you can invoke the item 'Screenshot' from the APPLICATIONS → ACCESSORIES menu in the top panel. In the resulting *Take Screenshot* window, select the desired options and click ML on the button labeled 'Take Screenshot'. Select the area if necessary[63] and then, in the resulting pop-up window, specify the desired folder and filename and click ML on the button labeled 'Save'.

8 Publishing Documents

Some of the programs installed in the Computational Physics Laboratory facilitate the preparation of technical documents, including documents with figures and elaborate equations.

8.1 TEX, LATEX, `dvips`, `xdvi`, and REVTEX

The programs TEX, LATEX, `dvips`, and `xdvi` for working with technical manuscripts are publicly available for almost any computer platform in existence and are described in Appendix A in *CPSUP*. Detailed information and downloadable files can be located by following links from the home page—`www.tug.org`—of the TEX Users' Group.

A better route for preparing documents that conform to the style of the American Physical Society involves making use of the APS-supplied add-on to LATEX called REVTEX, currently in version 4.1, which is fully compatible with LATEX2$_\varepsilon$. A copy of the *REVTEX 4 Author's Guide* and a copy of the *REVTEX 4.1 Author's Guide* are kept in a notebook in the CPL.

8.2 `pdflatex` and `ps2pdf`

The programs `pdflatex` and `ps2pdf` are routinely included with the standard LATEX distribution. When invoked with a statement like

[62] The definition of "active" might vary depending on how your desktop is configured.

[63] Move the cursor to one corner of the area, depress and hold ML, move the cursor to the diagonally opposite corner of the desired area, and release ML.

`pdflatex` *LaTeXFileName*

where the filetype `.tex` will be assumed, `pdflatex` will produce a PDF file directly from a LaTeX source file, though it does not correctly incorporate PostScript and encapsulated PostScript figures into the PDF document.

The second program `ps2pdf` takes as input a PostScript file created by `dvips` and creates a PDF output file containing the same document. This route properly incorporates PostScript and encapsulated PostScript figures into the PDF document. A statement to `ps2pdf` will have the form

`ps2pdf` *PostScriptFileName*

Here, the file type `.ps` or `.eps` must be included in the filename.

8.3 `pgf` and `tikz`

The LaTeX packages`pgf` and `tikz` provide an impressively elaborate and versatile means to generate a wide variety of graphics displays from within a LaTeX document. As such, learning its capabilities, especially its more sophisticated capabilities, will require substantial effort. Typing the command[64]

```
texdoc pgf
```

at a *Shell* window to your operating system will probably bring up links to a number of manuals, including the main—and voluminous (1100-plus pages!)—manual in the file `pgfmanual.pdf`. A brief orientation to the use of these packages is included in the Appendix A in CPSUP.

8.4 `aspell`

The program `aspell`, which is well on the way to replacing its predecessor `ispell`, is a general purpose spell checking program which is launched to check a particular file with the statement

`aspell check` *FullFileName* or `aspell -c` *FullFileName*

When used to check a LaTeX source file (file type `.tex`), standard LaTeX commands will not be flagged as misspelled. Full information about the use of `aspell` is available in the program's `man`-page.

8.5 Tgif

The program `tgif`, an interactive two-dimensional drawing tool that allows the user to draw and manipulate objects—rectangles, ovals, rounded-corner rectangles, arcs, polylines, polygons, open-splines, closed-splines, text—in the X-Window system is described in Appendix B of *CPSUP*.[65] Two versions of `tgif` are currently installed in the CPL. Version 4.1.43 is the older version which has simply been copied from the previous hardware in the CPL; version 4.2.5 is a newer version. The alias `tgif` will launch the newer version; the alias `tgifold` will launch the older version. Either can be followed by a file name. Backward compatibility of the newer version with files created by the older version has not yet been confirmed. If you are adventurous, you can try opening existing files in the newer version and reporting the results to the author. Otherwise, existing files should probably be opened with the older version.

[64] The designation `pgf`—Portable Graphics Format—reflects the name of the engine underlying the entire system.

[65] TGIF is copyrighted ©1990–2003 by William Chia-Wei Cheng, who grants to the user a non-exclusive, royalty-free license to copy, display, and distribute without charge, and to produce derivative works. The full copyright notice appears in the referenced appendix. Further information is available at the URL `bourbon.usc.edu/tgif/`.

9 Surfing the Net

The main program in the CPL for surfing the net is Firefox. To access Firefox in Linux, click ML on the "Web Browser" icon located on the top panel (if you have created that icon), select 'Firefox' from the INTERNET submenu of the APPLICATIONS menu, or type the command

```
firefox   or   firefox &
```

at the prompt from the operating system in a *Shell* window.

Firefox has an enormous number of opportunities for customization. By default, the program establishes two cache storage areas. One—the *memory* cache—is created at the beginning of a session, is effective throughout that session, and is erased when the session is terminated. The other—the *disk* cache—survives from session to session. Both speed the process of re-examining items already retrieved. With Firefox, user-customization allows control only of the size of the disk cache, and the memory cache is dynamically allocated, i.e., it re-sizes itself automatically. By default, the size of the disk cache has been set to 0 KB.[66,67] These values give reasonable "retention" of retrieved documents *during* a session but do not preserve retrieved documents from one session to the next.[68]

While continuing use may reveal the need for additional ground rules, two in particular apply to our collective use of web browsers:

1. *Please do not set the* disk *cache away from 0 KB. If everyone who uses the CPL adopts even a modest sized disk cache, in the aggregate quite a bit of disk space will be taken up. Almost always, disk space is better used for other purposes.*

2. *Unrestrained downloading of files from everywhere and anywhere can rapidly erode disk space. Certainly, if you find something that looks useful, acquire it to explore it, but then please remove it once it is of no further use. Let's not find ourselves—for long, anyway—with multiple copies of generally useful shareware. If the item is worthwhile, please talk with Mr. Cook about installing it in a public library so it can be available to all with a minimum consumption of disk space.*

The Lawrence home page can be accessed at `www.lawrence.edu`. The departmental homepage can be found by migrating to `www.lawrence.edu/dept/physics` and then opening the drop-down menu by clicking ML on the box labeled "Physics Site Menu" part way down the page before you.

10 Software Available/Terms of License

The software available to students within the CPL at Lawrence University is described in this section, which includes not only identification and brief description of each package but also information about the nature of the license that Lawrence has negotiated with the vendor and the way in which the software package is launched for use. Fuller information will be found in various chapters in *CPSUP*, in online books and `man`-pages, and in the documents supplied by the vendors of the software. Further contact information for the several vendors is compiled in Appendix Z of *CPSUP*.

[66] You can verify this setting by launching Firefox, migrating to the EDIT ⟶ PREFERENCES menu, clicking ML on the icon labeled 'Advanced' along the top of the *Firefox Preferences* window, and selecting the 'Network' tab. The box labeled 'Override automatic cache management' should be checked and the box labeled 'Limit cache to' should be set to zero. If that is not the case, please adjust whatever is in those positions to these values.

[67] You might take a few moments to explore other options in the *Firefox Preferences* window.

[68] Long-term storage of web pages in a disk cache is actually risky, since subsequent access to those pages will read them from the cache, despite the possibility that the page at the web site may have changed in the interim.

10.1 IDL

IDL[69] is an elaborate program for creating and processing arrays, processing images, and producing graphical displays, including animated displays. In the CPL, IDL is stored in the directory[70]

```
/apps/idl/idl85/
```

Because the aliases `idl` and `idlde` have been defined to point to the appropriate executable modules and necessary environment variables are set automatically when you login, IDL is invoked with the simple statement[71]

`idl` (for a command-line interface) or `idlde` (for the development environment)

at the shell prompt. Typing `exit` at the IDL prompt will terminate the session.

An introduction to IDL will be found in Chapter 2 in (the second edition of) *CPSUP*; more sophisticated features are described in the later chapters on solving ODEs, evaluating integrals, and finding roots, and in several volumes of IDL documentation available in the CPL. Further, simply typing a question mark at the `IDL>` prompt brings up a convenient IDL help window.

The Department of Physics at Lawrence owns licenses for twenty-five simultaneous uses of IDL. Twenty-one of these licenses are available in the CPL. The terms of the license restrict use to the machines in the CPL. Note that opening two IDL sessions on one machine will take two licenses.

10.2 MATLAB

MATLAB[72] is a competitor to IDL and, like IDL, is an elaborate program for creating and processing arrays, processing images, and producing graphical displays, including animated displays. In the CPL, MATLAB is stored in the directory[73]

```
/apps/matlab/matlab2012a/
```

Because the alias `matlab` has been defined to point to the appropriate executable modules and necessary environment variables are set automatically when you login, MATLAB is invoked with the simple statement

```
matlab
```

at the shell prompt. Typing `quit` at the MATLAB prompt will terminate the session.

An introduction to MATLAB will be found in Chapter 3 in (the second edition of) *CPSUP*; more sophisticated features are described in the later chapters on solving ODEs, evaluating integrals, and finding roots, and in several volumes of MATLAB documentation available in the CPL. Further, simply typing the statement `help` at the MATLAB prompt `>>` brings up a convenient MATLAB help window.

The Department of Physics at Lawrence owns two licenses for the Linux version of MATLAB.

[69] Harris Geospatial Solutions, Broomfield, CO, URL: `www.harrisgeospatial.com/Software-Technology/IDL`.

[70] This base directory for IDL changes every time a new version is installed. To find out the current IDL base directory, type the statement `echo $IDL_DIR`. The symbol `$IDLHEAD`—see Appendix B—used in *CPSUP* also points to this directory and must be replaced by the full path whenever it is quoted from *CPSUP*.

[71] On a PC, the IDL development environment for version 8.2 is launched by migrating through Start→Programs→IDL 8.2→IDL. The command-line interface can be launched by migrating through Start→Programs→IDL 8.2→Tools→IDL Command Line.

[72] The MathWorks, Inc, Natick, Massachusetts. URL: `www.mathworks.com`

[73] This base directory for MATLAB changes every time a new version is installed. To find out the current MATLAB base directory, type the statement `echo $MATLAB_DIR`. The symbol `$MATLABHEAD`—see Appendix B—used in *CPSUP* also points to this directory and must be replaced by the full path whenever it is quoted from *CPSUP*.

10.3 OCTAVE

While it is not installed anywhere in the CPL, a shareware program called OCTAVE is not quite as versatile as MATLAB but is syntactically almost identical to MATLAB. OCTAVE is freely available for many different platforms and, in particular, could be installed by any student on his or her personal computer at no cost.[74]

An introduction to OCTAVE will be found in Chapter 4 in (the second edition of) *CPSUP*; more sophisticated features are described in the later chapters on solving ODEs, evaluating integrals, and finding roots. Further, simply typing the statement `help` *CommandName* at the OCTAVE prompt `>>` brings up a description of the specified OCTAVE command. Typing the statement `doc` at the OCTAVE prompt `>>` launches a separate window giving access to a substantial document describing everything there is to know about OCTAVE.

10.4 PYTHON

PYTHON[75] is an elaborate shareware program for creating and processing arrays, producing graphical displays (including animated displays), and manipulating data in a variety of ways. It is available at no cost for a wide variety of operating systems. In the CPL, pieces of PYTHON are stored in a number of different directories. The executables that launch the CLI and the GUI are both stored in `/usr/bin`, which is in the search path followed by the operating system, so PYTHON can be invoked with the simple statement

`python` (for a command-line interface) or `idle` (for the *Python Shell*)

at the shell prompt. Each of these routes to starting PYTHON brings up a window in which statements can be interactively submitted at the PYTHON prompt, which by default is `>>>`. If a PYTHON program already exists in the default directory from which you launch PYTHON, that program can be run with the statement

`python` *ProgramName, including file type, usually* `.py`

Exiting from PYTHON to the operating system can be accomplished by typing the command `quit()` at the PYTHON prompt.

An introduction to PYTHON will be found in Chapter 5 in (the second edition of) *CPSUP*; more sophisticated features are described in the later chapters on solving ODEs, evaluating integrals, and finding roots. Voluminous on-line documentation, some of which is identified in Chapter 5 of (the second edition of) *CPSUP*, is also available.

10.5 MAXIMA

Installed on only a few of the stations in the CPL, MAXIMA,[76] a large shareware program for manipulating expressions symbolically, is a descendant of one of the earliest computer algebra systems, specifically of MACSYMA, which originated in the late 1960s at the Massachusetts Institute of Technology.[77] As with other computer algebra systems, MAXIMA is capable of manipulating expressions algebraically, differentiating and integrating expressions, solving algebraic and differential equations, producing a variety of graphs, finding eigenvalues and eigenvectors, etc.

[74] See information at the URL `www.octave.org`.
[75] See information at the URL `www.python.org`.
[76] See the information at the URL `maxima.sourceforge.net/`.
[77] MACSYMA, which is no longer available, was a proprietary—and expensive—program but MAXIMA is freely available for many operating systems, including Unix, Linux, Windows, Macintosh, and even Android.

An introduction to MAXIMA will be found in Chapter 6 in (the second edition of) *CPSUP*; more sophisticated features are described in the later chapters on solving ODEs, evaluating integrals, and finding roots. Voluminous on-line documentation, some of which is identified in Chapter 6 of *CPSUP*, is also available.

10.6 MAPLE

MAPLE[78] is an elaborate program for manipulating expressions algebraically, differentiating and integrating expressions, solving algebraic and differential equations, producing a variety of graphs, finding eigenvalues and eigenvectors, etc. At the time of the printing of this guide, Maple 16 was installed, but updated versions are installed as they are available. To facilitate your use of these tools, the aliases

`maple`	Start command-line interface for Maple 16
`xmaple`	Start standard worksheet for Maple 16

have been defined.

When you invoke the the standard worksheet in Maple 16 for the first time, you will be asked to choose between the "Document" interface and the "Worksheet" interface, with the Document interface being the default if you make no selection. These two interfaces are described in the screen that offers you the choice. You may wish to explore the Document interface, but the Worksheet interface will be easier to use—and will be more like what is described in *CPSUP*. Once you have made the choice by selecting one or the other interface with the mouse, that choice—and lots of other default parameters—will be recorded in the file `/home/`*YourUserName*`/.maple/16/maplerc`. This "rc" file will control the interface you see in all subsequent sessions. (You can change many of the parameters by selections available in the panels of the Options submenu of the Tools menu in the menu bar of the standard worksheet.)

Within that Worksheet interface, you may want to toggle the display of input lines away from the default so that the input lines displayed on the screen are instead by default identical to what you type. To do so,

(a) Launch the gui interface with `xmaple`,
(b) Select 'Worksheet' interface,
(c) Select the 'Display' panel of the Options submenu of the Tools menu in the menu bar of the worksheet interface,
(d) Toggle the selection labeled 'Input Display' from '2-D Math Notation' to 'Maple Notation',
(e) Click ML on the button 'Apply Globally' at the bottom of the *Options* window, and
(f) Select 'Exit' from the File menu.

This change will not affect lines already displayed in your worksheet but will make sure that all subsequent input lines in the current session will be displayed exactly as you type them and, because you applied the change globally, this form for input lines will become the default in subsequent sessions as well.

The Department of Physics at Lawrence owns eleven licenses for MAPLE. Documentation on the use of MAPLE will be found in Chapter 7 in (the second edition of) *CPSUP* and in the later chapters on solving ODEs, evaluating integrals, and finding roots, in various MAPLE manuals and books, and in the help menus available once the program has been invoked.

In the UNIX environment, the system-wide MAPLE initialization file is named `init`, is stored in the directory `$MAPLEHEAD/lib`, and can be created or edited only by the system manager. A user-specific initialization file, which *can* be created and edited by individual users, is named `.mapleinit`

[78] Waterloo Maple Software, Waterloo, Ontario, Canada. URL: `www.maplesoft.com`

and is stored in the user's home directory. When MAPLE is started, the system-wide file is executed before the user-specific file, so the user-specific file can override features created by the system-wide file.

10.7 *Mathematica*

Mathematica[79] is an elaborate program for manipulating expressions algebraically, differentiating and integrating expressions, solving expressions as differential or algebraic equations, producing a variety of graphs, finding eigenvalues and eigenvectors, etc. *Mathematica* is started by typing the statement

```
mathematica
```

in a *Shell* window, which brings up a *Mathematica* screen in which, among other options, you can select to open a new notebook or open an existing notebook, into either of which statements to *Mathematica* can be typed. Our use of *Mathematica* piggy-backs on a 10-seat site license owned by Lawrence. Thus, maximally ten simultaneous users can be supported throughout Lawrence. In some terms, courses offered in the Department of Mathematics and Computer Science make extensive uses of these licenses, so users in the CPL may encounter a "busy" signal when attempting to start up the program. In other terms, use by students in mathematics and computer science is minimal and students in the CPL may have freer access to the program. Documentation on the use of *Mathematica* will be found in Chapter 8 in (the second edition of) *CPSUP* and in the later chapters on solving ODEs, evaluating integrals, and finding roots, in various *Mathematica* manuals and books, and in the help menus available once the program has been invoked.

In the UNIX environment, system-wide *Mathematica* initialization files are named `init.m` and are stored in the subdirectories of the directory `$MATHEMHEAD/Configuration`. User-specifiable initialization files, which *can* be created by individual users, are named `init.m` and are stored in subdirectories of the directory `.Mathematica` that will be created in the user's home directory the first time *Mathematica* is started. When *Mathematica* is started, the appropriate system-wide file is executed before any user-specific file is executed, so the user-specific file can override features created by the system-wide file.

10.8 Numerical Recipes

The Numerical Recipes library is a commercially available package[80] that contains numerous FORTRAN and C subroutines and demonstration programs, each of which must be copied into the user's default directory and possibly edited before being compiled to serve the user's ends. In contrast to most licenses, which authorize using software on particular processors, Numerical Recipes licenses screens. Before the stickers became unusable, a sticker attesting to the authorization for Numerical Recipes was affixed somewhere to each licensed screen. Because we at Lawrence have purchased nine licenses for use of Version 2.10[81] of the FORTRAN recipes and nine licenses for use of the C recipes, each licensed screen had two such stickers, one authorizing use of FORTRAN versions on that screen and the other authorizing use of C on that screen. Copying of Numerical Recipes routines to locations that would permit their use on other screens than the licensed ones and use of Numerical Recipes routines from rooms in the residence halls via `telnet` or `ssh` access to a machine to which a licensed screen is attached are both violations of the terms of the license to which Lawrence has agreed.

An introduction to the use of Numerical Recipes is contained in Chapter 10 in (the second edition of) *CPSUP* and more detailed descriptions and examples are discussed in later chapters on

[79] Wolfram Research, Inc., Champaign, IL URL: `www.wolfram.com`

[80] Available from Numerical Recipes Software, Cambridge, Massachusetts. URL: `www.nr.com`.

[81] The current version is Version 3.x.

solving ordinary differential equations, evaluating integrals, and finding roots. The full library is described in the books published by Academic Press and identified at the end of the introductory chapter mentioned in the first sentence of this paragraph.

10.9 ODEPACK

ODEPACK[82] is a large package of FORTRAN subroutines for solving a wide variety of sets of ordinary differential equations. The full package, which includes the most elementary component, LSODE, has been installed at Lawrence. The use of LSODE is described in Chapter 11 in (the second edition of) *CPSUP*. The associated files themselves are stored in the directory `/apps/odepack`.

10.10 MUDPACK

MUDPACK[83] is a large package of FORTRAN subroutines for solving a wide variety of elliptic partial differential equations. The full package, which includes `mud2sp.f` and `mud3sp.f` for solving *separable* two- and three-dimensional partial differential equations, has been installed at Lawrence. The use of `mud2sp.f` and `mud3sp.f` is described in Chapter 16 of (the second edition of) *CPSUP*. The associated files themselves are stored in the directory `/apps/mudpack`.

10.11 Gzip, Gunzip, Zcat and Relatives

Running particularly under UNIX operating systems but sometimes available with other operating systems, the programs Gzip, Gunzip, and Zcat are standard, public-domain programs for file compression, file decompression to another file, and file decompression to a screen display, respectively. More specifically,

- Gzip, invoked with the statement `gzip` *filename*, *replaces* the specified file with a more compact file named *filename*`.gz`.
- Gunzip, invoked with the statement `gunzip` *filename*, *replaces* the specified file with the expanded file named by stripping the `.gz` from the input filename. Actually, the file can be specified either by giving its full name (e.g., `abcdef.gz`) or by giving the name *without* the file type `.gz`, which will be assumed by the program if it is not there. Indeed, for compatibility with previous versions, the current version will also accept file names whose file type is `.z`, either explicitly stated or left as an (unspecified) default.
- Zcat, invoked with the statement `zcat` *filename*, displays an unzipped version of the specified file on the screen but leaves the original zipped file intact on the disk. This program accepts filenames in the several formats described for Gunzip.

Help in using these programs will be displayed on the terminal by the statements '`gzip -h`', '`gunzip -h`', or '`gzcat -h`' (or '`zcat -h`').

Another set of programs, specifically `bzip2`, `bunzip2`, `bzcat`, and `bzip2recover`, for file compression is installed in the CPL. This family uses a different—and apparently more efficient—compression algorithm. Files compressed with `bzip2` will have file type `.bz2`. Details are available in the `man`-page for `bzip2`.

A third, now obsolete (and no longer available in the CPL), set of compression programs once was part of the UNIX operating system. The members of this set were `compress` and `uncompress`.

[82] Written by Alan C. Hindmarsh of the Computing and Mathematics Division of the Lawrence Livermore National Laboratory. URL: `www.netlib.org`

[83] Written by John Conway Adams at the National Center for Atmospheric Research in Boulder, CO. URL: `www2.cisl.ucar.edu/resources/legacy/mudpack`

This set of programs used the file type `.Z`—not `.z`—to indicate that the file is a compressed file. The format of the compressed file, however, seemed to be the same as that used by `gzip` and its siblings because files created by one set seemed (with the change of file type) to work with the other. Thus, if you happen to encounter a compressed file with file type `.Z`, try changing the file type to `.z` or `.gz` and expanding it with `gunzip`.

11 Computational Languages

FORTRAN-77, FORTRAN-90, FORTRAN-95, C, and C++ compilers are installed in the CPL. In this section, we illustrate how FORTRAN and C programs are compiled, linked, and run. A brief introduction to *writing* programs in these languages is contained in Chapter 9 in (the second edition of) *CPSUP*.

11.1 Compiling and Running FORTRAN Programs

The command `f77` is used to compile and load FORTRAN-77 programs. Suppose that the file `laplace.f` contains a FORTRAN program that does not draw on any functions and routines beyond standard FORTRAN. Then the simple statement

```
f77 laplace.f
```

will compile the program and link it to necessary system routines, storing the (binary) executable code in a file named `a.out` in the default directory. The statement[84]

```
./a.out
```

will then execute that program. We mention two alternatives:

1. The statement

   ```
   f77 -c laplace.f
   ```

   will suppress the linking step, producing only the compiled (and not yet runnable) file which will be named `laplace.o`. The statement

   ```
   f77 laplace.o
   ```

   will then effect the linking to produce the runnable file named `a.out`

2. The statement

   ```
   f77 -o laplace.xf laplace.f
   ```

   will compile and link the program `laplace.f`, but will place the resulting runnable program in the file `laplace.xf` rather than in the file `a.out`.

[84] The default directory is not in the search path on the HP machines. Thus, to execute a binary file in the default directory on the HP machines, the command *must* start with '`./`'.

All the details you would ever want to know will be found in the `gfortran`[85] `man`-page. There are also several books on FORTRAN in the CPL library.[86]

For practice, you might copy the file `/usr/share/CPSUP/lg/laplace.f` into your own directory, look at it with `more` or `gedit`, compile and run it to obtain its output, and maybe even try to figure out enough FORTRAN to modify the program so that it computes a solution on a finer grid or for different boundary conditions.

The solution printed on the screen when the program is run is, of course, unintelligible. You can obtain a better (graphical) sense of the solution by writing the output to a file with the statement

```
./a.out > laplace_f.dat
```

invoking IDL[87] and then issuing to it the statements

```
IDL> openr, 1, 'laplace_f.dat'      ! Open file
IDL> soln = fltarr(15,15)           ! Create array for solution
IDL> readf, 1, soln                 ! Read in solution
IDL> surface, soln                  ! Display solution as mesh surface
IDL> device, decomposed=0           ! Set sensitivity to color table
IDL> loadct, 3                      ! Load a particular color table
IDL> shade_surf,soln                ! Display solution as a shaded surface
IDL> exit                           ! Exit from IDL
```

Alternatively, once the data file has been created, you can create the graphical display by invoking MATLAB[88] and then issuing to it the statements

```
>> id = fopen( 'laplace_f.dat', 'r' );  % Open file
>> soln = fscanf( id, '%f', [15,15] );  % Read in solution
>> status = fclose(id);                 % Close file
>> soln = transpose( soln );            % Transpose solution
>> mesh( soln );                        % Display solution as mexh surface
>> surf( soln );                        % Display solution as a shaded surface
>> quit;                                % Exit from MATLAB
```

Alternatively, once the data file has been created, you can create the graphical display by invoking OCTAVE[89] and then issuing to it the statements

```
>> id = fopen( 'laplace_f.dat', 'r' );  # Open file
>> soln = fscanf( id, '%f', [15,15] );  # Read in solution
>> status = fclose(id);                 # Close file
>> soln = transpose( soln );            # Transpose solution
>> mesh( soln );                        # Display solution as mexh surface
>> surf( soln );                        # Display solution as a shaded surface
>> quit;                                # Exit from OCTAVE
```

[85] The FORTRAN compiler installed in the Fedora 17 operating system is actually `gfortran`, and the more common commands `f77`, `f90`, and `f95` are provided through links of those commands to the single command `gfortran`.

[86] *FORTRAN 77 Programming (Second Edition)*, T. M. R. Ellis, Addison Wesley (1990); *Introduction to FORTRAN 90*, L. Nyhoff and S. Leestma, Prentice-Hall (1999); *FORTRAN 90/95 for Scientists and Engineers*, S. J. Chapman, McGraw-Hill (2nd ed., 2004); *FORTRAN 90 for Engineers and Scientists*, L. Nyhoff and S. Leestma, Prentice-Hall (1997); *FORTRAN 90 Programming*, T. M. R. Ellis, I. R. Phillips, and T. M. Lahey, Addison-Wesley (1994).

[87] Possibly by typing `idl` at the shell prompt, but see the *Local Guide* for more detail.

[88] Possibly by typing `matlab` at the shell prompt, but see the *Local Guide* for more detail.

[89] Possibly by typing `octave` at the shell prompt, but see the *Local Guide* for more detail.

Alternatively, once the data file has been created, you can create the graphical display by invoking PYTHON[90] and then issuing to it the statements

```
>>> import matplotlib.pyplot as plt      # Import needed modules
>>> import numpy as np
>>> from mpl_toolkits.mplot3d import Axes3D
>>> f = open('laplace_f.dat', 'r' )      # Open file for reading
>>> soln = []                            # Initialize list for data
>>> for line in f:                       # Read, split each line
        soln.append( \
        [float(x) for x in line.split()] )
>>> f.close()                            # Close file
>>> data = np.array( soln )              # Convert list to array
>>> xx = np.linspace( 0.0,1.0,15 )       # Create arrays for independent
>>> x,y = np.meshgrid( xx, xx )          #    variables
>>> fig1 = plt.figure()                  # Create figure
>>> ax1 = plt.axes( projection='3d' )    # Specify 3D axes
>>> ax1.plot_surface( x, y, data )       # Create internal surface plot
>> plt.show()                            # Display plot on screen
```

The command `f90` or `f95` replaces the command `f77` if code that exploits the features of FORTRAN-90 or FORTRAN-95 is to be compiled and loaded. Note, however, that many of the features officially in FORTRAN-90 and FORTRAN-95 have, in fact, been incorporated as unofficial embellishments to the most recent version of FORTRAN-77, so the command `f77` may well compile some programs that in fact use features that are official only in FORTRAN-90 or FORTRAN-95.

11.2 Compiling and Running C and C++ Programs

The command `cc` is used to compile and load C programs. Suppose that the file `laplace.c` contains a C program that does not draw on any functions beyond standard C functions and routines. Then the simple statement

```
cc laplace.c
```

will compile the program and link it to necessary system routines, storing the (binary) executable code in a file named `a.out` in the default directory. The statement[91]

```
./a.out
```

will then execute that program. We mention three alternatives:

1. The statement

   ```
   cc laplace.c -lm
   ```

   will be necessary to effect the compilation and linking if the program uses mathematical functions like `sin`, `cos`, `exp`, etc. These functions are defined in the math library (named `m`), but the compiler does not recognize the need to link that library without help. Further, note that the option `-lm` must appear *after* any files for which its contents is needed.

2. The statement

[90] Possibly by typing `python` or `idle` at the shell prompt, but see the *Local Guide* for more detail.
[91] See footnote 84.

```
cc -c laplace.c
```

will suppress the linking step, producing only the compiled (and not yet runnable) file which will be named `laplace.o`. The statement

```
cc laplace.o
```

will then effect the linking to produce the runnable file named `a.out`

3. The statement

```
cc -o laplace.xc laplace.c
```

will compile and link the program `laplace.c`, but will place the resulting runnable program in the file `laplace.xc` rather than in the file `a.out`.

All the details you would ever want to know will be found in the `gcc` `man`-page and in three books in the CPL library.[92]

The `c++` command (Linux) replaces the `cc` command if C++ code is to be compiled. All the details you would ever want to know will be found in the `gcc` `man`-page.

For practice, you might copy the file `/usr/share/CPSUP/lg/laplace.c` into your own directory, look at it with `more` or `gedit`, compile and run it to obtain its output, and maybe even try to figure out enough C to modify the program so that it computes a solution on a finer grid or for different boundary conditions.

The solution printed on the screen can be written to a file and then displayed in IDL, MATLAB, OCTAVE, or PYTHON using statements similar to those in the penultimate paragraph of Section 11.1.

12 Accessing Other Machines

Two quite different methods for accessing remote machines are in common use. In the first, the user actually establishes a remote login to the distant machine. In the second, the user simply accesses the remote machine for the purpose of transferring files to or from that machine. We describe the first method in Section 12.1 and the second in Section 12.2.

12.1 ... for Remote Login

Two programs are available within the CPL for establishing remote logins on machines to which access is permitted. The older—and less secure—program is called `telnet`; the newer—and more secure—program is called `ssh`.[93] Statements like[94,95]

[92] *Applications Programming in ANSI C*, Richard Johnsonbaugh and Martin Kalin, MacMillan (1990); *The C Programming Language (2nd edition)*, Brian W. Kernighan and Dennis M. Ritchie, Prentice-Hall (1988); *C Programming*, Steve Worthington, Boyd and Fraser (1988).

[93] Because of its less secure character, `telnet` is slowly disappearing, and some potential hosts—including the CPL server NEWTON—will reject efforts to effect login with `telnet`. When `ssh` doesn't work, however, you may find `telnet` to be the only option.

[94] In the CPL, the statement `ssh` has been defined as an alias for `ssh -X`, which makes sure that `ssh` will allow the remote machine to create X-windows on the local machine.

[95] If the mnemonic name gives trouble, try substituting the numeric IP address of the desired remote host in the above statements. The statement '`nslookup` *Name*' will return the numeric IP address of *Name*, though it is not likely to work if `telnet`, `ssh`, or `ftp` have been unable to locate your desired destination. The numeric address for NEWTON is 143.44.10.21.

```
telnet NEWTON.phys.lawrence.edu
ssh    NEWTON.phys.lawrence.edu
```

which use the full identification of the remote host, or statements like

```
telnet NEWTON
ssh    NEWTON
```

which will also work because the file `/etc/hosts` provides a suitable translation from the shorter name to the full name, will initiate communication with a remote host. In most cases a username and password on the remote host will be necessary before the connection can be completed. Detailed information about each of these programs is available in its `man`-page.

Once established, a remote login with `telnet` or `ssh` will give you the impression that you are actually working in a *Shell* window on the remote machine. Tasks will be executed on that remote machine even though all screen displays will appear on the local machine. Of course, you must know something of the appropriate command structure on the remote machine before you will be able to accomplish any tasks on the remote machine.

On some UNIX machines, the statement `rlogin` *RemoteNodeName* will establish a remote session on the remote machine. Details can be found in the program's `man`-page.

12.2 ... for Transferring Files

The program `ftp` (file transfer protocol) is installed on all machines in the CPL for the purpose of transferring files from one machine to another. For illustrative purposes, one or the other of the statements

```
ftp    NEWTON.phys.lawrence.edu
ftp    NEWTON
```

will initiate a session on NEWTON, though NEWTON is directly accessible by other means from satellite nodes in the CPL, so you may need statements of this sort only when accessing machines more remote than NEWTON. As with `telnet` and `ssh`, login with a username and password will almost always be necessary.

After a successful connection to the remote machine, which will be indicated by a line that is something like

`230 User` *UserName* `logged in`,

you will be greeted by an empty `ftp` prompt, probably something like `ftp>`. From here, you are able to run commands that have many parallels with UNIX commands. The most important `ftp`-specific commands that you will need to know are

- `ls`

  Lists the files and directories stored in the working directory on the remote machine.

- `pwd`

  Displays the current working directory on the remote machine.

- `cd` *Path*

  Switches to a new default directory identified by *Path* on the remote machine.

- `put` *FileName*

  Copies the *one* file *FileName* in the current default directory *on the local machine* to a file of the same name in the current default directory *on the remote host*.

- `get` *FileName*

  Copies the *one* file *FileName* in the current default directory *on the remote host* to a file of the same name in the current default directory *on the local machine*.

- `del` *FileName*

  Deletes the file *FileName* from the current directory on the remote host.

- `binary`

  Forces `ftp` to transfer files as binary files. (By default, the transfer protocol is ASCII. Thus, to transfer files that are non-ASCII (such as Word documents, images, executable files, etc.), you will need to to toggle the binary flag to 'on'.

- `ascii`

  Restores ASCII mode for transfer of files.

- `help` *Command*

  Prints information about the command *Command*. Typing `help` without an argument will yield a list of all available commands.

- `bye` and `quit`

  Terminates the `ftp` session and returns you to a prompt from the operating system of the local machine.

You might want also to look at the help messages for `mput` and `mget`, which allow you to transfer more than one file (i.e., multiple files) to or from the remote machine, respectively.

A GUI interface to the `ftp` program is also available. Launched with the statement `gftp` or by selecting 'gFTP GUI Interface' from the Applications → Internet menu in the top panel, the program presents a window in which you can specify the particulars of the remote machine, identify the files to be transferred by selecting them with clicks of ML, and then request the transfer either from or to the remote machine. Information is available in the help messages within the program.

The Lawrence network is partitioned, and communication between the various partitions is tightly restricted. Restrictions prevent communication between the partition for University laboratories and the partition for student residences. It is therefore not possible for you to access machines in the CPL from your room in the residence halls.

Communication between an *on*-campus machine and an *off*-campus machine is also subject to constraints. Since a firewall isolates on-campus machines from the outside world, connections initiated by machines on our side of the firewall (Lawrence campus) and accessing machines on the other side of the firewall (the rest of the world) are handled in ways that are transparent to the user and the user need take no special action. Communication *into* Lawrence from off-campus locations is much more difficult. Transfer of files using `ftp` is complicated; actually accessing Lawrence machines via `telnet` or `ssh` *from outside Lawrence* is essentially impossible.

As an alternative to direct transfer of files using `ftp`, you can write the file on one machine to an off-line storage device (e.g., USB flash memory) and then physically transport that device for reading on the other machine. This method, which could be particularly useful in moving files between the machines in the CPL and your personal machine in your room, is described in Section 14.

Warning: Unfortunately, the windows operating system flags the endings of lines in text files differently than does the UNIX operating system, and strange characters may appear in the UNIX world if a Windows file is simply transferred "straight"—and *vice versa*. In many instances `ftp` knows how to deal with this wrinkle. The HP editors seem to know how to deal properly with files transferred from windows machines, even if they aren't properly translated by `ftp`. You may need to do some testing to make sure you end up with what you think you are getting at the destination.[96]

13 Mail

Because the workstations have been acquired to support mainly computational and graphical uses within the Department of Physics and as such should not be particularly visible to the world outside of Lawrence, we have made no particular effort to configure the email system in the CPL and it may well be unreliable or quirky. Indeed, because of the Lawrence firewall that guards against infiltration from the outside, an address other than the official University-assigned address on mail originating *outside* the firewall simply will not work. Internal nodes are completely invisible to the outside world. Only one route is available for accessing your University email from an HP workstation. To do so, bring up Firefox as a browser, go to the URL

```
http://webmail.lawrence.edu
```

and login in in the way you would if you were accessing your email from outside Lawrence.

14 Off-Line Storage

To help us keep disk usage under control, we ask that files that you wish to keep but that are not any longer of frequent use be saved on a USB flash drive. Standard USB flash drives will be recognized when plugged into a USB port and can be used for off-line storage or for transfer of files to other machines not accessible via `ftp`. Shortly after you plug such a device into a USB port on the HP machine, the operating system will detect and mount the device, an appropriately labeled icon will be added to the desktop and to those in the *DirView* window that is created when you double-click ML on the desktop 'Computer' icon. In addition, a *DirView* window displaying the directories and files on the USB device will appear on the desktop. Once this *DirView* window is displayed, you can then drag files to or from that window in the same way that you would move or copy files from one directory to another in your online directory structure. Equivalently, if you prefer, you can move and copy files to and from the USB device by using the *Shell* command `cp`. Note the (probably obscure) identifying name displayed at the top of the *DirView* window. If that name, for example, is `2D64-8AD7`, then that device can be accessed with the name `/run/media/`*YourUserName*`/2D64-8AD7`.

Warning #1: Before unplugging a USB device from a computer, remember to unmount it. On the HP machines in the CPL, this task is accomplished either (a) by selecting 'Eject' from the File menu in the *DirView* window, (b) by closing the *DirView* window, moving the cursor over the device's icon in the *Computer* window, and selecting 'Eject' from the menu that pops up when you press MR or (c) by selecting 'Eject' from the menu accessed by pressing MR with the cursor on the desktop icon for the drive. Whichever route you adopt, you need to wait a few seconds until the lights—if any—in your USB device stop flashing before unplugging the device from the USB port.

Warning #2: See again the warning at the end of Section 12.2.

[96] Files transferred with a flash drive *from* the HP world *to* the Windows world on a PC can be appropriately adjusted simply by opening the file in WordPad and then saving it again. Further, once so treated, the file can be transported back to the HP world and remains sensible without further adjustment.

Warning #3: Various innocent file types in the UNIX world may have particular applications programs associated with them in the DOS world. On some PCs, for example, files named `*.tex` will be interpreted as Microsoft Word files.

15 Miscellaneous Useful Features

In this section, we enumerate several features whose documentation is difficult to find in published sources but that effect useful actions.

15.1 Useful Statements

Some of the less common—but still useful—statements include

- `find . -name` *filename* `-print`

  This statement will search through the directory structure lying at and below the specified directory (here '.', the current default directory), look for a file by the name *filename*, and print (to the screen) the full path of any such files that it finds. Wild cards can be used in the statement but the file name must then be enclosed in single quotation marks to prevent these special characters from being misinterpreted by the `find` program. Detailed information can be found in the `find` `man`-page.

- `ps -ef`

  This statement instructs the computer to display a list of all processes currently running. The table

```
   UID   PID  PPID  C    STIME TTY      TIME COMD
  root   430     1  0 14:39:44 ?        0:10 /usr/Cadmin/bin/objectserver
 cookd  1142  1116 44 09:58:18 pts/1    0:00 ps -ef
  root   276     1  0 14:39:38 ?        0:01 /sbin/cron
```

  shows the header and a few selected lines from a possible output. Each line corresponds to a single process. The first entry (UID) in each line is the name of the user who owns the process; the second entry (PID) is a unique number identifying the process; the third entry (PPID) identifies the parent process, i.e., the process that launched the one corresponding to the line; and—ignoring the rest (see the `ps man` page)—the final item (COMD) identifies the statement and its arguments. The list may be long, and the user seeking a particular process may reduce its length by piping the output through `grep`, e.g.,

  `ps -ef | grep` *YourUserName*

  which will suppress the printing of any lines not corresponding to processes owned by the specified user. The most common use of this statement is to find the PID (not PPID) of a process that has gone astray so it can be terminated with the `kill` command, which requires the PID as its argument, e.g., `kill 1385`. (Users without privileges, of course, can kill only processes that they own.)

- `cpio ...` (copy in and out)

  While the command `mv` can be used to relocate a directory from one point to another on the *same* disk, the command `cpio` provides a more convenient means to transfer entire directories from one disk to another. With the default directory set to the head of the directory that is to be copied, we use `find` to search for every file in that directory and its subdirectories, output

each file *name* to the standard output device, and pipe that output as input to `cpio`, which copies each file thus identified to the desired end directory (which should be created before the process is begun). With the default directory properly set (with `cd`), we execute the statement

`find . -depth -print | cpio -pdvum` *DestDirect*

to copy the contents of the default directory and of all its subdirectories to *DestDirect*. The option '`.`' tells `find` to start at the current default directory, the option '`-depth`' instructs `find` to descend the directory tree from its starting point, and the option '`-print`' causes `find` to output each file name in turn to the standard output device. The options specified to `cpio` have the meanings

- '`-p`': read input from the output of `find`.
- '`-d`': create directories as needed.
- '`-v`': display a list of files copied on the standard output device. Standard UNIX file redirection could be used to send this list instead to a specified file.
- '`-u`': copy unconditionally, even if an older file will then replace a newer file with the same name.
- '`-m`': give copied file the file modification time of the original, not the time the copy is made.

The options '`-p`' and '`-d`' are necessary; the option '`-v`' is prudent to provide reassurance that the transfer is proceeding properly; the option '`-u`' is relevant only when copying files into directories that may already contain files; and the option '`-m`' assures that copied files will not appear to have been created more recently than they actually were. By default, `cpio` will *not* follow symbolic links, though that behavior can be overridden with the option '`-L`'.

- `which` *ProgramName*

  This statement instructs the operating system to display the full path to the (binary) file it will execute when *ProgramName* is submitted as a statement at the prompt from the operating system. For example, the statement `which pwd` will return the response `/usr/bin/pwd`.

15.2 Important Special Files

Customization of programs is often accomplished by the use of an assortment of special files. Some are created by the system manager and establish system-wide defaults, while others can be created by each individual user, are private to the creating user, and permit that user to adapt the behavior of assorted programs to that user's particular needs and preferences. We mention two routes to accomplish these customizations.

- `rc` files. Many UNIX programs can be customized by specifying options in a program-specific resource file (`rc` file) in the user's home directory. Each such file typically has a name that starts with a dot, continues with the name of the program, and concludes with the two letters `rc`. We have already met the `.cshrc` and the `.maple/16/maplerc` files. Many other programs besides these two look for such files and respond to them in various ways.

- The `.Xdefaults` file. When an X-window is opened by a particular application, the application often looks to a file named `.Xdefaults` in the user's home directory for information about how the window is to be positioned, sized, colored, etc. Programs following this convention respond to user customization as stipulated in this file. Since specifications for all programs are incorporated in the *same* file, it can become quite long. Whatever its length, each line has the general format illustrated by the template

  ProgramName`*`*ParameterName*`:` *ParameterValue*

where *ProgramName* is the name of the program (perhaps abbreviated), *ParameterName* is the identifier of an X-window parameter, and *ParameterValue* is the user-stipulated value for the identified parameter. More specifically, the line

```
Tgif*Foreground:  Navy
```

specifies that, in an X-window opened by the program tgif, the foreground color is to be navy. Some information about the defaults actually used by programs when nothing specific is included in the `.Xdefaults` file can be found by examining the files in the directory `/usr/share/X11/app-defaults`.

15.3 Shell Scripts

If a particular sequence of UNIX statements is frequently used, you may wish to construct a file containing the statements and then execute them by invoking the resulting *shell script* with a statement like

`source` *ScriptName*

In response to this (single) statement, the system will execute the statements in the script, one at a time in order. The user of the script is spared the task of typing each statement individually and manually every time the sequence is needed.

Shell scripts can contain any statements that might otherwise be entered at the prompt in a *Shell* window. Comments are flagged with the character '`#`', which can occur at the beginning of a line or anywhere in the line and which will cause the operating system to ignore the rest of that line in the file. Note, however, that the *first* line of the shell script must be a blank line, an executable statement, or a very special comment of the form

```
#! /sbin/sh
```

which specifies that the script is to be executed by a particular shell. Any other comment in the very first line of a shell script causes error messages.

16 Tips From Fellow Students

In this section, we enumerate several hints that we hope will facilitate your use of the *Shell* window, text editors, and other resources available in the CPL. Many of these items have already been addressed (or will be addressed) in other sections of the *Local Guide*. We elect, however, to mention them again to underscore their importance.

- Keyboard shortcuts

  UNIX operating systems often make keyboard shortcuts available and even allow users to create their own shortcuts. The subject is too vast for inclusion here of much more than an alert to the possibilities and a suggestion that you peruse `man`-pages; the options available when you open 'System Settings' from the menu in the toolbar headed by your name, click ML on the icon labeled 'Keyboard', and select the tab labeled 'Shortcuts'; and other resources to learn about these options. We mention two shortcuts in particular.

  - The TAB key

In typing long filenames or directories, the TAB key can be used as an auto-complete device. For example, if you are about to type the filename `/home/`*YourUserName*, you can usually just type the first few letters of the words `home` or *your username* and then the TAB key to fill the remainder of the word.[97]

- The ALT-TAB key

  If one of several windows on the desktop is on top and you wish to move another to the top, you can simply hold down the ALT key while pressing the TAB key a suitable number of times. On the first typing of the TAB key, a box containing icons for each window on the screen will appear; subsequent typings of the TAB key will cycle the highlighted icon through the possibilities. When you release the ALT key, the window corresponding to the highlighted icon will be brought to the top. (Of course, you can accomplish the same end alternatively by clicking ML on the corresponding button in the bottom panel.)

- Multiple Desks and Workspaces

  Using additional work spaces in Linux can prove to be very convenient, especially if you are going to have many windows open at once.

  - By default in Linux, there are four work spaces at your disposal. Each functions as an independent desktop within your account. In workspace 1, for example, you might be running IDL and gedit, and in workspace 2 you could have MAPLE and an extra terminal running. To select the desired workspace, hold down *both* the CONTROL key *and* the ALT key while typing the left or right arrow keys, ⟨CONTROL-ALT-$\longleftarrow$⟩ and ⟨CONTROL-ALT-$\longrightarrow$⟩, one or more times to cycle through the several work spaces.

    Once a particular workspace is active, you can create *Shell* windows, launch programs, etc. just as you would in the (one) workspace that was the default the first time you logged in. Further, while you cannot move icons around in the taskbar across the bottom of your desktop, you can move windows in one workspace to another workspace. Simply click MR on the icon you wish to relocate, move the cursor to the item 'Move to Another Workspace >', and select the desired new workspace in the resulting pop-up menu.[98]

- `cd ..`

  This variation on the commonly used command `cd` allows you to move up one level in the directory structure. If you were currently using the file `problem2.tex` in the directory

  `/home/`*YourUserName*`/comp.mech/chapter2`

  and you wanted to select a problem in another chapter by going back into your `comp.mech` directory, all you have to type is `cd ..` and you will move from the `chapter2` directory into the `comp.mech` directory. You could also use the `cd ..` command along with the new directory you want to enter. For example, to move from the directory `chapter2` into the directory `chapter1` (both subdirectories of the `comp.mech` directory), you could simply execute the statement

  ```
  cd ../chapter1
  ```

  After this shift has been effected, a simple `pwd` command will display the (new) current directory, thus verifying the action of the `cd` command.

[97] The auto-completion feature looks for uniqueness within the filename or directory path. For example, if there are two files named `comp_mech_2005_idl_01.pro` and `comp_mech_2005_maple_01.mws`, and we assume that they are the only two files in the current working directory, typing the characters "co" and pressing TAB will extend the filename to `comp_mech_2005_`, at which point the two file names become different and, without additional input, the computer doesn't know which file you actually want.

[98] This operation will work regardless of whether the window is minimized, maximized, or free-floating in GNOME.

- The importance of `&`

  The `&` is a very useful symbol in the *Shell* window. If you open up a text editor from the *Shell* window and do not use the `&`, then when the text editor opens you will no longer be able to type in the *Shell* window. If you wanted to use the shell window while having the text editor (or some other application) open, then you should use the `&` symbol after the command that starts the application. If. for example, you wanted to open the file `problem8.tex` in a text editor and still retain the ability to execute other statements in the *Shell* window, then (assuming that you are already in the correct directory), you would simply type the statement

  ```
  gedit problem8.tex &
  ```

  to launch the editor and detach it from the launching *Shell* window.

- Organization with Directories

  Although you will be working on computers *owned* by the Department of Physics at Lawrence University, your space on the user disk is truly *yours*. Using the computers to play games, download music, watch movies, etc. is strictly prohibited, but you should not be afraid to design your directory structure to your liking. In other words, don't be afraid to create directories, subdirectories, subsubdirectories, ... in order to organize the storage of your files. Recall that the statement

  `mkdir` *DirectoryName*

  is used to create a new directory. Creating separate directories for each course or project (and possibly subdirectories within those directories) will help keep you well-organized from the beginning and will help you find older files in the future for reference purposes.

- Selective Printing

  If you have a long LaTeX document and you only want to print a certain section of it, you can extract the desired pages from the `.dvi` file and send them to the printer with statements like

  ```
  dvips -o FileName.ps -pp 5-12,18,23-26 FileName
  lpdup FileName.ps
  ```

  Here, *FileName* stands for the name of the intended file *without* a file type; the command `dvips` will supply the default file type `.dvi` for the input file. Note (1) that there can be *no* spaces in the list of desired pages and (2) that page numbers identify physical pages from the beginning of the document and will coincide with the numbers on the pages only if the document starts with page 1 and does not reset the page numbers along the way.

- Reference guides in the CPL

  As you have probably already noticed, the CPL is equipped not only with computers but also with a small library of physics books and computer reference books as well. Do not hesitate to use the physics books, reference guides, manuals, solutions manuals, the book *CPSUP*, and, of course, this *Local Guide* often. These materials are in the the CPL to help *you*, so don't be afraid to use them.

- The importance of courtesy in the CPL

  The Rules of Citizenship in the CPL will be laid out in Section 17. We here want to emphasize the importance of a few of these rules/common courtesies. When you are finished working with particular references, it is courteous to tidy up your work area before you leave the CPL. If you are working on a big assignment and you are referring to reference guides, solutions

manuals, and class notes, then please try to return these items to the bookshelves when you are finished so that other students can find (and use) them easily.

Another way to tidy up your space is to log out whenever you leave the CPL for more than a few minutes, especially the night or two before a big assignment is due when the computers are in high demand. If you leave the CPL for a few hours and don't log out of your computer, then the person who comes in and sees that your computer is the only one not being used is in an awkward position. Does he or she log out of your computer for you, or wait until you return? Well, he or she will probably log you out, so hopefully you at least saved everything before you left. This is a big risk to take, so if you are leaving the CPL for more than a few minutes, save all unsaved files and log out.

- *ALWAYS shut the CPL door behind you when you leave.* All those who have legitimate reason to use the resources in the CPL have card access to the space. If you leave the door open, however, then everybody who has access to Youngchild and Steitz Halls has access. A few thefts from the CPL have occurred, but they are preventable. *Just close the door when you leave!*

- Some hints about using PRO-files in IDL

  PRO-files in IDL can be very useful, but you must be aware of where you are saving the PRO-files that you want to use in an IDL session. For example, if you plan on using two PRO-files in the directory `/home/`*YourUserName*`/comp.mech/chapter2`, then, to have access to those files from within IDL, you must make sure that that directory is the default directory before starting IDL.

  The first time you invoke an available function or procedure PRO-file in IDL, IDL will find and compile the PRO-file without problem. If, however, you then edit this file while IDL is still open, IDL will *not* discover that the file has changed, since IDL thinks that it already has a compiled function or procedure by the name you are using. You must alert IDL to your editing of the file. Merely save the new version of the PRO-file and then enter the command `.compile` *name_of_PRO-file*. This command will update the compiled version internal to IDL to reflect the new, saved version.[99]

- Some hints about MAPLE

  When using MAPLE, you may often be defining functions and variables and then redefining them. If you engage in such actions and then re-execute a statement that was executed sometime earlier in your session with MAPLE, you might find that you don't get what you expect because, in the activity between the first and second execution of the statement in question you changed the definitions for functions and variables and the new definitions rather than the old are being used in the second execution. One easy way to correct this problem is to insert the command `restart:` at the end (or wherever else needed) in the MAPLE coding. That way you can reset all of the values and re-enter specific commands anywhere in the code. However, you must be careful because important equations and variables are usually defined at the beginning of the code. In other words, you may need to execute several commands at the beginning of the code after the `restart:` command in order for everything to work properly.

  Deleting items in MAPLE can be a bit tricky. If you wish to delete a character in a command line, you simply use the ⟨BACKSPACE⟩ key. You can also delete whole commands this way. Portions of the display on the screen can be deleted by highlighting them and then typing either the ⟨BACKSPACE⟩ key or the ⟨DELETE⟩ key, but you will have to experiment a bit to find out exactly what you must highlight to effect the desired deletion. *Be aware, however, that this action only removes the items from the display on the screen; it does* not *remove variables defined by the deleted statement from MAPLE's behind-the-scenes workspace.*

[99] *Command* PRO-files—files executed with the statement `@`*FileName*—are *not* compiled and will be reinterpreted by IDL every time you invoke the file. In this case, edits to the file will be recognized the next time IDL executes the file.

- Some hints about text editors

  Each of the major text editors (`gedit` and `emacs`) has a menu in the tool bar called SEARCH. In this menu, there are two particularly interesting features, FIND and REPLACE. With FIND, you can find all instances of the word 'plot', for example. With REPLACE you can replace all instances of a word with another word. This capability is especially useful in the case of misspellings.

  Work with IDL is facilitated by opening a text editor (detached from the window in which you are running IDL), typing the statements you wish IDL to execute in the text editor, and then copy and paste them into the *Shell* window in which IDL is running. With this strategy, minor changes can be made to the statements without having to re-type *all* of them into IDL. Merely edit the statements in the text editor and then re-paste the whole block of statements into IDL.

  When working with a `.tex` file in a text editor, you may find it useful to move the statement `\end{document}` in order to locate errors more easily. For example, if you are adding a few very complicated equations in the middle of the document, moving the `\end{document}` statement temporarily to the end of each equation in turn and processing the document through LaTeX each time will (a) reduce the number of error messages you receive and (b) limit the region of the file in which you must search to locate the error(s) responsible for the message(s) you *do* receive.

17 Rules of Citizenship within the CPL

The CPL is a shared facility, and individual users must be respectful of the rights of other users. Further, certain precautions are necessary to protect the equipment. Finally, the space must be viewed as an upper-level physics laboratory, not a general-purpose, institution-wide computer room. In that latter context, only those who have accounts on the physics network (sophomore, junior, and senior physics majors and students outside the department who happen in the term to be taking an *upper-level* physics course) are entitled to use this laboratory. In this section, we summarize some of the guidelines that have been adopted to assure smooth use of the resources of the CPL. Some of these guidelines have already been mentioned earlier in this document.

Nearly 50 individuals are authorized to have access to the CPL. Given that number, and given increasing curricular demands that students use the resources of the CPL, adherence to these guidelines is especially important. Failure to follow these guidelines may result in denial of access to the CPL or locking of accounts.

1. The CPL is a space for quiet study and work with the computing resources, not a space for large group discussion of problem assignments or for socialization. Conversation among two or three individuals that is soft enough to respect the need of others in the room for quiet is, of course, acceptable, but anything more distracting than soft conversation is out of order. Discussions involving more than two or three persons should take place in other spaces (Y-104 or Y-115, to both of which your ID card gives you access, or the Atrium).

2. Monitor your disk usage and clean house frequently. Several times during the past several years, we have had moments when the user disk became quite full, most often because some large files have been kept longer than necessary. We have not yet implemented a system of disk quotas but may do so should courtesy and good sense among our users be found wanting. We ask that you remain aware of your disk usage and diligent in housekeeping so that files that are no longer needed, especially large ones, are deleted regularly. Secondary files and PostScript files produced by LaTeX and assorted log and backup files should not be kept long-term. Image files from the xray machine, the plasma laboratory, and other places can require

lots of disk space, perhaps not individually but certainly in the aggregate. Further, large `tar` files downloaded from remote sites should not be kept for long. Statements that may be useful in helping you to maintain control over your disk usage include

- `du -s` or `du -sk`

  which will report to you in kilobytes the total disk usage in the default directory and all of its subdirectories. Normally, you would want to execute this statement with the default directory set to your home directory.

- `du -s *` or `du -sk *`

  which will report to you in kilobytes your disk usage, file by file and subdirectory by subdirectory in the default directory.

- `rm` *FileName(s)*

  which will delete the specified files from your directory, though—by default—will ask for confirmation of your desire to delete each file before doing so. A full path can be specified with each file; in the absence of a path, the file will be sought in the default directory.

- `rm -f` *FileName(s)*

  which will delete the specified files *without* asking for confirmation.

- `rm -f *.bak`

  which will delete from the default directory *all* files having file type `.bak`, not asking for confirmation along the way.

- `rm -fR *`

  which will completely empty the default directory of *all* files and all subdirectories, including files and subdirectories in all subdirectories to all depths but leaves the (empty) default directory in place. This statement is incredibly dangerous, and should be executed only when you are *absolutely* sure you know what you are doing.

- `deltex`

  which is aliased to

  ```
  rm -f *.aux *.log *.dvi  *.toc *.ilg *.idx *.ind *.bak *.bck *.out
  ```

  and will remove all files that can easily be recreated from a saved `.tex` file or other primary file. This command does not ask for confirmation before deleting files.

- `rmdir` *DirectoryName(s)*

  which will delete the specified *empty* directories from your directory structure. A full path can be specified with each directory; in the absence of a path, the directory will be sought in the default directory.

- `find . -name '*.bak' -exec rm {} \;`

  which will search through the directory structure lying at and below the specified directory (here '.'—the current default directory), look for any file whose name ends in `.bak`, and then remove that file.

3. Dragging icons to the dumpster or the trash bin does not remove the associated files from the disk but merely relocates them to a different folder in your directory structure. These files continue to usurp space on the disk. Empty the dumpster or trash bin regularly by selecting 'Empty Trash' from the menu that pops up when you press MR with the cursor on the icon labeled 'Trash' on the desktop.

4. Occasionally, the system may crash when you are using it. With some crashes, the system manages to create a file named `core`, usually in the current default directory or in your home directory. That file is likely to be very large—it is essentially a memory dump—and it is a binary file, meaningless to any but the most dedicated of computer gurus. Anytime you find a file named `core` in one of your directories, please delete it without giving it a second thought.

5. Do not behave in ways that even begin to risk getting food, beverages, gum, and other undesirable items in or on any of the equipment in the CPL. In other words, *food and drink are prohibited in the CPL.*

6. Playing of CDs or other sound-making devices that anyone other than you can hear is also prohibited in the CPL. If you must have sounds in your ears in order to work comfortably, bring in your walkman and use earphones—and don't have the volume up so loud that the bass leaks out into the room despite your precautions. Use of the capabilities of the workstations themselves to play CDs constitutes an inappropriate use of equipment acquired for much more serious purposes.

7. Playing games on the workstations in the CPL is against the rules. It is an entirely inappropriate use of equipment acquired for much more serious purposes. The CPL is not a video arcade. Playing computer games almost inevitably leads to behavior that is alien to the spirit of the CPL.

8. Using the Linux machines for word processing of non-scientific text should be rare. Extensive use of these machines for web-based email and personal tasks is frowned upon. Furthermore, the use of the workstations as music- or video-downloading platforms, or for creating CDs not in the spirit of the CPL is expressly prohibited.

9. Unless you have a long job running on a particular workstation (in which case you should leave a sign telling other users so and predicting the time at which you will return to check on the job), log out when you leave the CPL. While all of our users are responsible and respectful of others, you nonetheless leave yourself vulnerable to inadvertent corruption when you leave without logging out.

10. Save your files frequently (and particularly when you leave for more than a minute or two). Unanticipated power failures or disruptions can result in loss of unsaved information. A *very* large part of the responsibility for protection against such losses rests with individual users.

11. When leaving a workstation, please log out (or leave a sign if you have a long job running), restore manuals to the bookshelves and remove your personal possessions so that the next user finds the area in the state in which you would like to find it when you come in to use the CPL. Please also take responsibility for maintaining a tidy appearance in the CPL, which is a showcase facility in the department. Viewing a cluttered space does not create the best impression on unanticipated important visitors, e.g., prospective students and their families.

12. *Never allow your workstation screen to become locked.* Make sure that the option to lock your screen automatically after some period of inactivity is turned *off*, and try to avoid inadvertently selecting the 'Lock Screen' option if it is even present in any of the menus in the top panel.

13. Be concerned about security; turn off lights and lock up if no one is there when you leave. Do not worry about locking someone else out. It is each user's responsibility never to leave the room without taking along his or her ID card.

14. Do not hesitate to log out an unattended machine that is not accompanied with an explanatory sign, especially if it is the only machine available when you want to use the CPL.

15. Do not hesitate to shut down and reboot a workstation if its screen is locked, especially if it is the only machine available when you want to use the CPL. Setting the desktop so that the screen locks after some period of inactivity is a violation of the Lawrence University honor code.

16. `Telnet`, `ssh`, and `ftp` access to the machines in the CPL from off-campus and from rooms in the residence halls is now blocked by the structure of the network. Such access *within* the CPL should be limited to short uses for the purpose of submitting a job to the printer or checking on the status of a running job. The user sitting at the keyboard of a particular workstation

has the right to expect that the *full* resources of that machine are available, and extensive telneting from rooms in the residence halls renders that expectation incorrect.

18 References

Additional information about the workstations and about the Linux operating system can be found in the `man`-pages and in several online tutorials and demonstration programs. Among the possibly useful books in the CPL are the following:

Brian W. Kernighan and Rob Pike, *The UNIX Programming Environment* (Prentice-Hall, Englewood Cliffs, NJ, 1984)

Stephen G. Kochan and Patrick H. Wood, *Unix Shell Programming* (Prentice-Hall, Carmel, IN, 1993)

Mark G. Sobell, *UNIX SYSTEM V: A Practical Guide* (Benjamin/Cummings, Redwood City, CA, 1995)

Margaret Levine Young and John R. Levine, *UNIX for Dummies Quick Reference* (IDG Books, Indianapolis, 1994)

A Hardware in the CPL

All satellite nodes in the CPL are HP Z620 workstations purchased and installed in the summer of 2012 and running the Fedora 17 implementation of Linux. Each has the following characteristics:

- Processor: 1 Intel® Xeon® Processor E5-2630 2.3GHz, 15 MB Cache, 1333 MHz Memory, 6 Cores, Hyper-Threading, Intel vPro Technology
- Chipset: Intel® C602 Chipset
- Memory: 6 GB 1600 MHz DDR3 ECC Unbuffered RAM
- Storage: 500 GB 7200 rpm SATA NCQ; SATA SuperMulti DVD+/-RW; Integrated SATA 6 Gb/s controller, RAID 0, 1, 5, 10 capable
- Audio: Integrated Intel/Realtek HD ALC262 Audio
- Ports:
  - Front: 2 USB 3.0, 1 USB 2.0, 1 IEEE 1394a standard, 1 microphone in, 1 headphone out
  - Rear: 2 USB 3.0, 4 USB 2.0, 1 audio in, 1 audio out, 1 microphone in, 2 PS/2, 2 RJ-45 to integrated Gigabit LAN
- Power: 800 Watt 90
- HP Compaq LA2205wg 22" flat screen monitor with two USB ports

The CPL also is equipped with two network printers (an HP P4015x Monochrome Laser Printer and an HP 4700dn Color Laser Printer).

Finally, the workstations in the CPL are all satellite to a server located in the Lawrence Library. In essence, the server has a single CPU with 1 GB of memory, a 100 GB hard drive as a system disk, and a 200 GB SCSI drive as the home drive. The server runs Red Hat Linux Version 5.5. Note, however, that the server is in large part a virtual machine, which means that, should one or more or its components fail, a component somewhere else in the Lawrence total computer system would more or less transparently take over and users would hardly notice the transfer.

IP addresses and node names for the components in the CPL are listed in the following table.

IP Address	Full Node Name	Short Name
	Server	
143.44.10.21	newton.phys.lawrence.edu	newton
	Workstations	
143.44.32.78	feynman.phys.lawrence.edu	feynman
143.44.32.79	landau.phys.lawrence.edu	landau
143.44.32.80	noether.phys.lawrence.edu	noether
143.44.32.81	einstein.phys.lawrence.edu	einstein
143.44.32.82	fermi.phys.lawrence.edu	fermi
143.44.32.83	brahet.phys.lawrence.edu	brahet
143.44.32.84	brahes.phys.lawrence.edu	brahes
143.44.32.85	leavitt.phys.lawrence.edu	leavitt
143.44.32.86	mayer.phys.lawrence.edu	mayer
143.44.32.87	yalow.phys.lawrence.edu	yalow
143.44.32.88	chandra.phys.lawrence.edu	chandra
143.44.32.89	dirac.phys.lawrence.edu	dirac
	Printers	
192.168.188.84	CPL_HP4200.phys.lawrence.edu	
192.168.188.85	CPL_HP4700.phys.lawrence.edu	

B Local Translation of Generic Symbols in *CPSUP*

Throughout *CPSUP*, several symbols are used to refer to paths to various directories in the local environment. These symbols, with their generic meaning and their specific translation in the CPL, are listed in this table.

Symbol	Translation at Lawrence (LINUX world)	Significance
`$IDLHEAD`	`/apps/idl/idl85`	Head of the IDL directory structure.
`$HEAD`	`/apps/CPSUP`	Head directory in which files supplied with *CPSUP* are stored.
`$MAPLEHEAD`	`/apps/maple/maple16`	Head of the MAPLE directory structure.
`$MATHEMHEAD`	`/apps/Mathematica`	Head of the *Mathematica* directory structure.
`$MATLABHEAD`	`/apps/matlab/matlab2012a`	Head of the MATLAB directory structure.
`$MUDHEAD`	`/apps/mudpack`	Head of the MUDPACK directory structure.
`$MAXHEAD`	`/apps/maxima`	Stores links to the executables for the MAXIMA GUI and CLI.
`$NRHEAD`	`/apps/numrec`	Head of the Numerical Recipes directory structure.
`$OCTAVEHEAD`	`/apps/octave`	Anticipated head of OCTAVE directory structure; not yet implemented.
`$ODEHEAD`	`/apps/odepack`	Head of the ODEPACK directory structure.
`$PYTHONHEAD`	`/apps/python`	Anticipated head of PYTHON directory structure; not yet implemented.
`$TEXHEAD`	`/apps/texlive/2012`	Head of the TEX/LATEX directory structure.
`$TGIFHEAD`	`/apps/tgif/tgif-4.2.5`	Head of the directory structure for Version 4.2.5 of TGIF.
`$TGIFHEAD`	`/apps/tgif/tgif-4.1.43`	Head of the directory structure for Version 4.1.43 of TGIF.
`$USERHOME`	`/home/`*UserName*	User's home directory.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z