# Never Whistle While You're Wizzing

Lab Assistant Manual For 6.001 Wizards

by
Natalya Cohen
and the Scheme Team

# Warning

As you might've guessed from the title, the author of this document has been recently exposed to a very dangerous, mind-disturbing lloigor of a book, co-authored by Robert Shea and Robert Anton Wilson. Thus, some of the references in this manual will appear slightly amusing to the illuminated ones and obscure to the mehums (see what I mean?). As a matter of fact, most of it is plain gibberish, so don't let the silly remarks distract you into ignoring the serious and important stuff contained in the manual. (What can I say? Writing an LA manual, of all things, is such an unilluminizing experience that you almost have to introduce a factor which makes it worthwhile. Just be grateful I did not employ the writing style of the authors of "The Illuminatus! Trilogy.") For what it's worth, though, I hope I've intimidated you into reading the book, and I also hope this manual proves to be useful otherwise, as a guide to being a successful Fnord! Lab Assistant.

# 1 Administrivia

Welcome, the Legionnaires of the Lab of Dynamic Discord[1]. Following is a quick review of the administrivia—your basic duties outside the lab, as well as the rules of lab coverage. You will probably be brainwashed enough about this in LA meetings.

## 1.1 Timecards

Timecards are picked up in the Course VI administrative office, 38-445. This is also where you go at the beginning of the term (regardless of whether you LAd the term before or not) to fill out the forms necessary to get you on payroll. Timecards should be filled out and signed by the LA and dropped off in a pendaflex folder in the top drawer of the filing cabinet in the lab by the end of the day on Friday (if you have lab hours on weekends, go ahead and claim them ahead of time, as long as you plan to make them). The head TA or LA will sign them and turn them in for you by 10:00 the following Monday morning so that you can get paid the following Friday.

The anal-retentive administrators downstairs request that you indicate and circle the total number of hours you worked, as a check on their addition skills. Also, you should sign your timecards, not in the place supplied for the head TA/LA signature, but to the right of that space, under the lower right hand corner of the time table.

In addition, for the purposes of monitoring Fnord! your work more effectively (although the Eye in the Pyramid is always watching[2], the head LA will check over your timecards to make sure the hours you claimed are reasonable—just in case), please make sure your timecard makes it clear what the numbers on it represent. Specifically, write LAB to the right of the hours you spent in the lab, MEET by the hour of the LA meeting, and PREP by the hours of preparation—i.e., doing the problem sets and checking over the solutions. If you were covering the lab hours for someone else, you might want to indicate the name of that person as well. On the weeks of the quiz reviews, write REVIEW by the hours you spent giving the review.

[1] Otherwise known as Little Deluded Dupes.

[2] If you don't believe me, look at the back of a dollar bill.

As the above makes clear, an LA is paid not only for lab coverage, but also for attending the LA meetings, doing the problem sets, and, occasionally, giving the quiz reviews.[3] Each of these will be discussed in the following sections.

## 1.2 Lab Hours

Time slots for lab duties are selected at the first Lab Assistant meeting, at the beginning of the term. The sign-ups are traditionally done by seniority, based on the number of terms as a Lab Assistant (LAs that have served the same sentence receive priority based on their class year), although usually everyone is reasonable enough that any serious conflicts can be worked out quickly and easily without too much bloodshed. Once the lab hours are assigned, they are your responsibility for the rest of the term. The head LA keeps the schedule of the lab coverage; it is also put on the blackboard on the back of the lab.

Be prompt. It is always a good idea to show up ten to fifteen minutes ahead of time and spend the extra time making sure nothing and nobody is dead, catching up on the new developments, assessing the situation, meditating, etc.—see Chapter 2 for details.

Be prepared. This means you should be familiar with the problem set the students are working on to the extent that you can help them understand it and catch their bugs—and that, you will find, requires a pretty good preparation.

If you find that you have a permanent schedule conflict, try to find another LA who can either take your hours or trade with you, talk to the head LA, or bring it up in the LA meeting.

Similarly, if you cannot make a shift, it is your responsibility to get a replacement. *Please* start looking for one early, if at all possible. You can either get someone to "koppel"[4] your hours or trade them. This is relatively easily accomplished by sending out e-mail several days before your shift, but last minute cancellations are usually more of a problem. **If you cannot find anyone who would koppel your hours**, let the head LA *and* the head TA know. There have been precedents of graduate TAs and even lecturers coming in to help out.

[3] Sometimes, you may get paid for baking chocolate chip cookies or even watching the head TA play hockey—but we'll leave it to you to figure out how to finagle the head TA into letting you do that.

[4] In honor of Josh Koppelman, who claims to have taken a lot of people's hours on demand.

**Don't ever punt your lab hours.** Your lab hours are your responsibility. If you punt them, you are shafting hordes of 6.001 students who are eager for your guidance and advice, and bringing dishonor to the 6.001 staff, the EECS department, MIT, and the entire quadrant of the galaxy. You will be hunted down, dealt with, and dogs will eat your bones.

If a last-minute distraction should arise, for example, if you are kidnapped by space aliens on your way to the lab, be sure they let you make a phone call to the head TA or lecturer and leave a message reporting the emergency.

In other words, be very responsible about this. Everybody has problems from time to time, and nobody expects you to be perfect, but you *are* expected to arrange for someone to cover your hours at all times, even if you are feeling particularly imperfect.

## 1.3 LA meetings

The weekly LA meeting (with the head TA, and sometimes one of the lecturers, present) is scheduled at the beginning of the term, hopefully to fit into everyone's schedule. As mentioned before, you get paid for this hour long meeting (needless to say, you don't get paid if you don't show up). Make every attempt to attend all of the meetings. If there's a conflict, let the head LA and/or head TA know.

In the meeting, the upcoming problem sets (and sometimes solutions) are handed out and discussed. Also, this is your chance to give the rest of the 6.001 staff your feedback on how things are going in the lab and what specific problems students are having with the material.

## 1.4 Preparation

You also get paid for no more and no less than two hours for preparation every week a problem set is given out (the early ones usually take much less than two hours, the late ones take longer, so it all balances out). Draft problem sets are given out to you in the meeting at least a week before they go out to the students. Do every problem set (not just glance at it, really *do* it) as soon as it is handed out to you. This is not only to prepare you for your lab hours, but also to catch bugs and judge the difficulty of assignments. You will normally be given draft solutions as a reference (not a crutch) which you should also eventually read over carefully so we can catch bugs in these too. Send your bug reports to `6001-feedback@sicp-00.mit.edu`.

If you missed the LA meeting where the problem set was given out, some extra copies are usually available in the top drawer of the filing cabinet. If not, you can always get one from the course secretary.

At the beginning of the term, course notes, a copy of this manual, the "Don't Panic" manual, Revised 4th Report on Scheme (R4RS), and two floppy disks are given out to every LA. Read the manuals at least briefly. If you want to keep the problem set solutions you hack up in the lab, use your floppies to store them; do not leave your code in some random temporary directory you created.

## 1.5 Quiz Reviews

LAs are usually responsible for organizing and executing review sessions before the quizzes (and sometimes the final). In the past, these have been done in many different ways. Usually, the course material is divided into sections which are then allocated to an LA or two, and the allocated LAs prepare a few examples of problems which demonstrate the ideas in their section. Preparation for the review takes place of otherwise cancelled prep hours for that week (no problem sets are due on the week of the quiz), and the hours you spend doing the review can be claimed as well.

## 1.6 Mailing Lists

You should be prepared to check your mail *a lot*—at least, once a day during the week. You will need it to stay up-to-date. Besides, since you will be on a couple of mailing lists listed below, not checking your mail may prove fatal—for example, if your mailbox lives on Anathema, you may go over your quota. And you really don't want to read through thirty–forty–fifty 6.001-related messages *alone* at a time.

Following are the e-mail addresses of the major cliques and conspiracies behind 6.001. All the bugs you (or students) find as well as reports of the problems you find within the system should be sent to `6001-feedback`. Matters of relevance to the LAs are addressed through `6001-las`; please don't use `6001-staff` for this purpose, but do use it when you wish to illuminize the entire staff on some issue. Those are probably the only three you'll ever use; the rest are there for reference.

`6001-lecturers@ai.mit.edu`
: the lecturers

`6001-secretary@ai.mit.edu`
: the course secretary

`6001-recitation@ai.mit.edu`
: the recitation instructors

`6001-tas@ai.mit.edu`
: the graduate TAs

`6001-head-ta@ai.mit.edu`
: the head TA

`6001-head-la@ai.mit.edu`
: the head LA

`6001-las@ai.mit.edu`
: the Little Deluded Dupes themselves

`6001-staff@ai.mit.edu`
: the entire staff: lecturers, recitation instructors, TAs, LAs, and course secretary

`6001-feedback@sicp-00.mit.edu`
: people who should know about/are responsible for fixing bugs; includes lecturers and LAs but also people in charge of the system and the lab, for example; this is also the mailing list the students use to report bugs and register complaints—it may get quite hectic at times

# 2 In the Lab

This chapter is about your basic duties in the lab, assuming no crisis occurs. For the ways to deal with minor, as well as major, crises refer to Chapter 4.

## 2.1 What to Do When You Get to the Lab

If you believe in the Second Law of Thermodynamics, you will believe me when I say there are enough things to check over when you start your shift. That is why it is advisable to show up a little earlier, especially if your hours are on a busy night. In any event, here are the things you should worry about when you first get to the lab.

### 2.1.1 The Snakes

Make sure that all the snakes are functional. Logout or reboot machines where necessary. If rebooting doesn't help, or some other problem is encountered that prevents the snake from working correctly, follow the procedure for reporting hardware problems described in the Chapter on crises—i.e., inform the instrument desk people, send mail to `6001-feedback`, and put a tag on the machine (the tags can usually be found on top or in the top drawer of the filing cabinet). If you encounter a machine with a tag on it, but you do not recall reading about the problem on `6001-feedback`, follow the first two steps—it is better to report the problem twice than to neglect it altogether.

When first checking the snakes, have a look at the network lights on each machine to make sure it isn't swapping without any apparent reason (the two network lights are the ones with arrows above them; they will be blinking heavily if the machine is swapping). You should also do this from time to time as you walk around the lab. A machine should never be swapping heavily, even when in use. If it is swapping, either some joker is logged into it over the network, or there are a few abandoned Schemes also running on the machine and need to be killed (see section 4.2.1).

If you have absolutely nothing better Fnord! to do, one useful and heroic practice is to go around killing old Scheme processes. This might not be a problem anymore, but it proved to be a rather painful annoyance in the past. Hail Eris.

The two servers, sicp-00 and sicp-01, are in the closet behind the printers. You will certainly notice it if either one of them is down. Pray to the Goddess and refer to the chapter on crises. All Hail Discordia.

### 2.1.2 The Printers

The two printers, sicp-48 and sicp-49, are to the left of the entrance into the lab (sicp-49 is the left one). Make sure both of them are in working condition and on-line (the ON LINE light should be on), there are no paper jams anywhere, and there is enough paper. Throw old printouts into the recycling bin.

If either printer is low on paper, get some from the instrument desk (if it is open) or from the server closet. Resort to visiting the closet only if the instrument desk is closed. In this case, send e-mail to the `6001-las` to warn the first person who comes in the next day to re-supply the closet cache.

When getting paper for the printers, make sure there are always a couple of extra reams in the closet. (If the desk folks give you any grief, tell them to grow up.)

### 2.1.3 The Rest of the Lab; Lost and Found

Make sure the filing cabinet is locked whenever you're not using it. It often contains problem set solutions, and we don't want them to be of public use.

The Lambda Lounge, or the tutorial area, should be a pleasant place to be in. Straighten out the furniture. Get rid of all the junk people throw in there. Erase the irrelevant stuff from the blackboard. **Water the plants** if it looks like they need it—we don't want them dead.

Speaking of junk, you might want to tour the lab, looking for old printouts to recycle and other things people tend to leave behind. If you find something that appears useful—e.g., someone's floppy—throw it in the "Lost But Found" box by the door. If the object is really valuable—e.g., an expensive HP calculator—lock it in the filing cabinet and put a sign on the board to that effect, so that the person can recover it from an LA when (s)he comes in.

If you want to be totally compulsive, you may also go around arranging the chairs neatly by the tables. It is incredible what other things you can think of when bored—like having a Bavarian Fire Drill. A pity it doesn't happen that often.

### 2.1.4 Settling into your Shift

Add your name to the list of "LAs On Duty" on the blackboard and erase any names of LAs who are not on duty (erase your own name when leaving the lab). Read the '`motd`' file and/or announcements on the back blackboards to see if there is anything new, and erase any outdated messages. If the help queue is on, start helping the students listed on the queue.

If the lab is not busy, you will probably spend most of your time in front of a terminal. In that case, it is a good idea to put an LA sign on top of your machine so that students can see you[1], especially if the help queue is off. Don't forget to remove the sign when you leave.

## 2.2 The Help Q

The help queue is kept on the blackboard to the right of the Lambda Lounge. In fact, it is a good idea to keep at least two separates queues—one for people with technical problems, usually dead or hung up Schemes, where your help is needed in recovering the students' work; the other for people with conceptual problems. Priority is given to the first of the above.

You are the one who decides if the queue is on or off. You definitely want it on if the lab is busy, to avoid (further) chaos. During the more quiet hours, handle the queue however feels most comfortable to you. Make sure to check the ON or OFF box on the Q board.

When the queue is on, help students in the order their machine names are listed on the board[2] (watch out for people who sign up twice, just in case they will have another question by the time you get to them again—this is a common phenomenon on the more busy nights). The map of the lab with the machines numbered is posted on the wall by the board. **Always erase the name of the machine off the Q before going to help the student**, so that if there are any other LAs on duty, they can proceed to help the next person on the queue. If you get distracted while in the process of going over to someone's machine (having already dequeued the machine name), be sure to remember to come back to it afterwards—if you are afraid you may forget the machine name, write it down somewhere.

When the queue is off, the students should look for an LA and ask for help. Beware that some students will tend to use the queue even when it's off. Help them but explain Fnord! what the

---

[1] At least, until we get the LA baseball caps.

[2] At the beginning of the term, some will put their own name instead of the machine name on the queue. Straighten them out.

policy is. To bring the point even closer to home, put a big cross or some other hideously noticeable sign across the board[3]—the simple OFF box doesn't seem to do the trick.

And the last important point: if no LA has hours scheduled immediately after yours, turn the queue off in the manner described above when leaving the lab. Otherwise, the LA coming in a couple hours after you might find a stale queue waiting for him/her (if you do come in and find that the queue is a few hours old, don't hesitate to erase it; students should know that signing up for help when there is no LA in the lab, in hopes that one will come around, is not only useless but not a sociable thing to do).

## 2.3 Using and Updating Student Resources

These are the resources available to 6.001 students in the lab. People will be asking you about these, so you should definitely know about all of them. You should also refer the students to these when they ask you questions which can be answered with the help of '`motd`' or `M-x info`, for example.

THIS IS IMPORTANT: whenever you change the message of the day, the info tree, the default in the '`probset.scm`' file, or anything else which needs to be changed *on all the machines*, you have to make the relevant changes **to both servers**. See Chapter 4 for details.

### 2.3.1 On-Line Documentation

The info tree installed in the lab contains the on-line version of the "Don't Panic" manual, this manual, MIT Scheme Reference Manual, MIT Scheme User's Manual, Emacs documentation, etc. You can activate the info tree by typing `M-x info` in Edwin (type `C-h m` for available commands). Familiarize yourself with the contents of the major sources, so you know where to refer students later. In particular, you should read the "Don't Panic" manual, which contains the answers to most of the questions people ask in the first couple of weeks of the term. Rather than spending the time explaining something to the students, tell them to read the manual first (RTFM!).

You will notice that one of the menu items in the tree is "Food." Considering the food Fnord! policy in the lab, you might wonder what it's doing there. After you figure that one out, look under the entry named "Taco Bell Delivery" for more surprises.

[3] You may resort to drawings, in which case the Sacred Chao symbol could be used, or to obscure signs of the sort "Anyone who uses the queue while it's off will be beheaded. THE MGT."

Another useful feature is an Emacs/Edwin tutorial, accessible by typing `C-h t`. Refer students to this if they don't have much experience with Emacs and need to learn the basic commands.

### 2.3.2 Message of the Day

'`Motd`', or message of the day, file exists in the '`u6001`' home directory. It contains corrections/bugs in the current problem set students should know about, as well as some relevant announcements about problem sets/quizzes/lab hours/etc. If you login as u6001, you will see the contents of this file in the '`*motd*`' buffer on login. Since '`motd`' changes all the time, you should make sure you know what the '`motd`' file says at all times, for that is the primary source of bug reports.

If you or a student encounters a bug in the current problem set, you should (besides reporting the bug to `6001-feedback`, as usual) change the '`motd`' file to reflect that. The file is writeable by hacker, so you can easily edit it from Emacs (or Edwin) if logged in as hacker (or if you open an Emacs window while su'd to hacker). To open an Emacs window, if logged in as u6001, type `emacs &` from an xterm; if logged in as hacker, choose the appropriate label after clicking on the left mouse button. The '`motd`' file, as well as the blackboard, is also a good place to put the answer to a commonly-encountered problem or question.

### 2.3.3 Changing the Default of `M-x load-problem-set`

The problem sets are installed on the machines by the head TA or the head LA. However, it is the responsibility of the LAs to make sure the number of the current problem set is the default in `M-x load-problem-set`. Specifically, if a problem set was due on Friday and a new problem set has been handed out, the change should be made by the LA who is first to come in on Saturday.

To change the default, go to the file called '`~u6001/psets/probsets.scm`', where you will find something like the following definition:

```
(define problem-sets
  '(4                          ; this number is the default problem set
    (1 (load "ps1go.scm"))
    (2 ... )
  ... ))
```

As it stands, the default is 4. To change it to 5, simply change the car to 5, leaving the cdr in peace (the file is, again, writeable by hacker and can be edited in Emacs or Edwin).

### 2.3.4 Blackboards

For some reason, some students seem more comfortable reading the bug-related information and other kinds of announcements that are typically put in '`motd`' off the blackboard. For this reason, and also to make sure people who logged in *before* the most recent change in the '`motd`' file are well informed, you should probably duplicate the most important messages on the blackboards at the back of the lab.

Another type of thing to put on the board is, say, an Emacs issue students seem to be having trouble with, something that is likely to remain on the board for a long time, such as "Use `C-x C-f`, not `C-x b`, to create new files."

The schedule of lab coverage is put on one of the boards at the beginning of the term, to be consulted by LAs and/or students.

### 2.3.5 Filing Cabinet

The key to the filing cabinet can be obtained from the instrument desk. If the desk is closed, an extra key may be found in the server closet, on the chain hanging by the light switch.

The top drawer of the filing cabinet contains extra copies of the problem sets and solutions. If a student requests a copy of a current (or previous) problem set, give it to him only after making sure it isn't a draft version (these usually say DRAFT in big letters on the top). If there are no copies of the legitimate problem set available in the lab, students should try the 6.001 tutorial area (36-115) or the office of the course secretary.

The top drawer also contains such other useful things as a stapler, which students tend to ask for a lot, extra floppies, etc. A few copies of "Don't Panic" and of the text itself, as well as other references (including such winning documents as the Scheme Reference Manual) can usually be found in the "Head LA" drawer—these are more of a benefit to the LAs than the students. Oh, and if you are inside the "Guano" drawer, don't knock out Mr. Potatoehead [sic.]. Other drawers are less interesting.

Again, **lock the filing cabinet** after you are done using it.

### 2.3.6 Sign-up Sheets

Putting up sign-up sheets is a responsibility of the head LA. You should familiarize yourself with the sign-up rules posted below the sheets, however. On busy nights, people might try to cheat by signing up for more than one slot at a time. If you notice this, or if a student complains to you about this, you should Fnord! act according to the rules.

### 2.3.7 Drop Cards

Drop cards printed on the spot! Explore one of the nicer features of the system, `M-x print-drop-card`.[1]

## 2.4 Passwords, Codes, Locks

Besides the lab door codes, you have to remember the server closet door code and the hacker password. Both of these will be given to you at the first LA meeting. Please remember them (don't rely on them being written down or saved somewhere, like your mailbox), since you will be pretty much helpless without them in times of crisis, and use them wisely.[2]

Even more importantly, don't let the students find out what they are. Knowing the hacker password allows students to copy each others' work, not to mention the damage they can do to the system. A little knowledge is a dangerous thing.

Be *super* careful when opening the closet door (yes, this means pressing your body against the door and covering everything that remains to be seen with a palm of one hand while pushing the

[1] Actually, it would be nice if someone would extend it to be even more of a feature.

[2] *You* are more trustworthy than all the Buddhas and sages.

buttons with the other). Remember: the closet contains the two servers, the key from the filing cabinet, the printer paper, and more. If some well-meaning curious student gets in there, we're toast!

## 2.5 Other Lab Policies

### 2.5.1 Food

There is absolutely, positively no food or drinks allowed in the lab. Whenever you see a person come in with food or in the process of eating something, say in a very serious, official tone of voice "Excuse me, but there's no food in the lab. Please finish eating it right now. And... can I have some?" Better plots will wait until we get a big picture of Dr. Professor Hal Abelson with a piece of pizza in one hand and a bottle of Coke in the other to hang on the wall.

### 2.5.2 Frisbees

There is absolutely, positively no frisbee throwing allowed in the lab, except when done by LAs (not on duty) on Thursday nights. (There are a few nice glow-in-the-dark frisbees, courtesy of Burger King, stored in the "Kipple" drawer of the filing cabinet.) Posting of the signs to this effect will be postponed until more frisbees are obtained to hang underneath the signs.

### 2.5.3 Phone

If you are not helping someone, answer the phone (no, only if it rings, dupe). The list of possible responses is hanging on the wall, by the phone. The standard one is "00 Lab." The more creative ones include "Hello, I'd like to order a pizza," "This phone booth reserved for Clark Kent," "What?" "Yo, Idaho!" "Are you a turtle?" and, of course, "Where the hell is DML?"

By the way, in case *you* ever need to call the lab, the phone number there is x3-7976. The thing to do when calling (unless you are actually trying to accomplish something) is to wait for some clueless person to answer with "001 Lab." If that happens, respond with "Oh, sorry, I wanted the 004 Lab," and hang up. Ones and fours can be switched appropriately.

### 2.5.4 Alarm

If the room alarm starts whining, turn it off. Basically, you have to wave your hands a little by the door to the lab. But don't just do it; make it dramatic and look important. This works especially well when you're on rollerblades, but—oh, did I forget to mention?—there is absolutely, positively no rollerblading in the lab, and there's even a $25 fine.

# 3 Software in the Lab

There are only a few user accounts on the system. The two you should worry about are u6001 (with the inconspicuous password u6001, as the dialogue box readily reveals), which all 6.001 students use, and the hacker account, existing mainly for the purpose of unwedging students, the password to which will be given to you in the first LA meeting and as it changes. We will first discuss the student band, as you will spend most of your time dealing with it, then talk about the hacker account and privileges.

## 3.1 General Software Setup

When you login as u6001, an Edwin window (Scheme running Edwin) is automatically started for you. You will be doing most of your work inside the Edwin window, although opening an xterm might prove useful, especially when unwedging people. You can open an xterm (and, also, xclock and xload) by pressing the left mouse button and selecting the appropriate label.

As any respectable system would have it, there are three levels of abstraction you should worry about. At the bottom level is the underlying UNIX (also, the floppies use MS-DOS format). On top of that is bare Scheme, the manifestation of which is the so-called Edwin xterm window in the bottom right hand corner of the screen. Finally, there is the student Scheme environment, running Edwin (that is, your Scheme/Edwin process is the child of the process running bare Scheme—you can verify that by typing `ps -ef` in an xterm and observing the results). Edwin itself is written in Scheme. We will discuss the things you should know about each of the levels starting from the top.

## 3.2 Student Scheme/Edwin

The "Don't Panic" manual is a good source if you want to familiarize yourself with the student environment. In this section, you will find things that are not covered in the manual or those which we want emphasized.

### 3.2.1 Scheme Debugger

**Learn how to use the debugger.** As an LA, you will spend a majority of your time debugging students' code, so you should be comfortable with the Scheme debugger. Refer to "Don't Panic" for initial info. You will find (as soon as you stop whining that noone has taught you how to use the debugger and just try it) that it really is quite easy to use. For starters, write and try to execute a buggy procedure, answer `y` when prompted by Edwin on whether you want to enter the debugger, and type `C-h m` to get a list of commands.

You will find that sometimes—whenever a student's code runs on top of a big chunk of source code, as is often the case in problem sets involving graphics, for example—the debugger is not very helpful, since it mostly shows you what happens to the expression being evaluated as the source code, not the code the student wrote, is being run. The debugger is particularly useless if the source code is compiled, and it usually is. That is probably not a feature, but there is not much you can do about it. Try to get *some* information out of the debugger, but, if all fails, resort to some other method of debugging.

Encourage students to learn to use the debugger as well. Most of them have an inexplicable paranoia of it Fnord! which overtakes them and forces them to type `n` whenever the friendly Edwin tries to help out. Show them how easy it is; just a simple `C-h m` goes a long way. Teaching the students some of the basics will ultimately save you a lot of work, not to mention that it will deny them another excuse to use you as a human debugger and force them to be more independent.

## Things to Look for when Debugging

While we are on the subject of debuggery: you will find that, especially at the beginning of the term, students will be generating many predictable, classifiable bugs. Infinite loops, for example, are usually rather obvious. Another common bug in the beginners' code is an unused definition (e.g., writing an internal iterative procedure without ever calling it). Later in the term, there will be quite a number of students wedging themselves by not being careful with their parentheses. An easy way to fix that is to always start your debugging by going to the beginning of the definition and typing `C-M-q` (next section describes this and other indentation commands). You don't want to spend half an hour staring at a piece of code only to realize that some parens were in the wrong place. Seems rather obvious when dealing with a LISP-like language, but an amazing number of people needs to be reminded of this particular feature of Scheme.

There are also some more obscure, but still rather common, error messages. "Anonymous arithmetic error" means a piece of compiled code tried to do some arithmetic and suddenly lost; it's usually caused by division by zero. If you find something else that pops up a lot, bring it up—there are usually different common bugs associated with different problem sets.

### 3.2.2 Useful Edwin Commands

Edwin is just a version of Emacs written in Scheme. Thus, most of Emacs commands work in Edwin. Here we remind you of the most useful ones (students usually neglect most of these and resort to the more primitive means of achieving their goals, which really is a pity; you might try to open the eyes of the more clueless ones and save them and, consequently, yourself a lot of time and frustration). They should be tatood upon the eyelids of the more clueless students.

## Motion

`C-n, C-p, C-f, C-b`
: Use these to move up (previous), down (next), forwards and backwards. Don't use the losing arrow keys.

`C-a, C-e`
: Moves the cursor to the beginning/end of the line.

`C-M-a, C-M-e`
: Moves the cursor to the beginning/end of the definition.

`M-<, M->`
: Moves the cursor to the beginning/end of the buffer.

`C-v, M-v`
: Scrolls to next/previous screen.

## Multiple Windows

`C-x o`
: Switches you to another window. You *do not* need your mouse to do this!

`C-x 2, C-x 5`
: Splits the current window in two vertically/horizontally.

`C-x 1`
: Deletes all windows but the current one.

`C-x 0`
: Deletes the current window.

`C-x ^, C-x {`
: When used with a numeric argument, allows you to extend the length of a vertically-split window/width of a horizontally-split window by a number of lines indicated; negative argument makes it shrink.

## Buffers, Files, Directories

`C-x b` Switches you to another buffer after prompting for its name (the default is the latest visited buffer). If the buffer doesn't exist, creates it. Some students get wedged when they use this command to create new buffers, since they expect a file under the same name to be created as well—this is not the case. Tell them to use C-x C-f instead.

`C-x C-b` Lists all buffers.

`C-x C-f` Finds a file (hit `TAB` to see your choices at any point); if the file doesn't exist, creates it in the current directory. This can also be used to get the directory listing—just hit `RET` after the name of the directory.

`C-x C-s` Saves the current buffer to a file under the same name. If the file doesn't exist, asks you for the name of the file to write to.

`C-x C-w` Saves the current buffer, prompts for a name of the file to save to.

`C-x d (M-x dired)`
Produces the directory listing. Hit `C-h m` for the commands in this buffer. Also, a short listing of files in the directory can be accessed by issuing `C-x C-d`.

## Search and Query Replace

`C-s, C-r` Prompts you for a string to forward/backward-search for (the default is the last thing you searched for). To continue the search, press `C-s` or `C-r` again. The students usually look for things by going up and down and up and down using the arrow keys—which is *not* very efficient.

`M-x query-replace`
Replaces one string by another. You will be given an opportunity to either make the change (press `SPC`) or not (press `n`) at every occurence of the string. Type `!` to replace *all* the occurences.

## Indentations

`TAB` Correctly indents (pretty-prints) the current line with respect to the previous line (another use of `TAB` is for completions; see below).

`C-M-q` Indents each line of the following form (s-expression). This is an extremely useful command, since you can usually tell whether the parens are matched properly just by looking at the indentation.

`C-M-f`, `C-M-b`
: Finds the matching parenthesis to the one the cursor is currently on. This, as you might imagine, is also quite helpful.

## Killing and Yanking

`BS` Kills the character to the left of the cursor; does not enter it on the kill ring. Students *love* using it because it is so primitive. Don't wait for them for half an hour to delete a couple of lines like this; tell them about `C-k`.

`C-d` Kills the character the cursor is currently on; does not enter it on the kill ring.

`C-k` Kills everything on the line to the right of the cursor, putting the deleted text on the kill ring so it can be yanked later.

`C-SPC` Sets the mark.

`C-x C-x` The exchange point and mark command (point is where the cursor is). Shows you where the mark is.

`C-w` Kills the marked region. To mark the region, set the mark at the beginning of the region with `C-SPC` and go to the end of the region.

`M-w` Enters the region onto a kill ring without removing it from the screen; used to copy text as opposed to cutting and pasting.

`C-y`, `M-y` `C-y` yanks the last piece of text on the kill ring (the text you deleted by the means of either `C-k`, `C-w` or `M-w`). *After* typing `C-y`, you can keep trying `M-y` to recall the text killed earlier.

## Completions

`TAB` Completes command names/filenames/etc. in the minibuffer; if several completions are possible, displays the choices available. Again, students usually ignore this and type `M-x load-problem-set`, for example, while, in fact, `M-x lo TAB p TAB` will achieve the same result; illuminize them, please.

`M-TAB` When used in the '`*scheme*`' buffer, completes the names of procedures, both primitive and the ones currently defined in the global environment (actually, you can use this inside the debugger, too, in which case it is not the global environment, but the environment of the frame you are currently in—winning, isn't it?); also displays available choices in case of ambiguity.

## Evaluating Scheme Expressions

`C-x C-e` Evaluates the expression just before the point regardless of where it is in relation to the parentheses. I advise you to use this instead of `M-z`.

`M-z` Evaluates the current definition (searches backwards for a left paren on the left margin, then searches forward to find its mate, evaluates the expression in between).

`C-M-z (M-x eval-region)`
Evaluates a marked region.

`M-o` When used in a scheme-mode buffer, evaluates the entire buffer (any buffer with a name that ends in '`.scm`' is in scheme mode by default; otherwise, use `M-x scheme-mode` to change the mode to scheme mode). Cannot be used in the '`*scheme*`' buffer.

`M-p` When used in '`*scheme*`' buffer, yanks the latest expression evaluated. May be used over and over to recall previously evaluated expressions.

## Printing

`M-x print-buffer`
Prints the current buffer; the default printers are sicp-48 for machines sicp-02 through sicp-24 and sicp-49 for the rest. To change the default (in case one of the printers isn't working), type `M-x set-variable RET lpr-switches RET '("-Psicp-49")` to print to sicp-49, for example. To print without a header page, make that `'("-h -Psicp-49")`.

`M-x print-graphics`
Prints the picture displayed in one of the graphics windows. You can specify which window to print from by either giving its name (what it is bound to in your Scheme) or clicking on it (the default option).

`C-q C-l` Quotes `^L`, which forces a page break on that line. `C-q` in general is used to quote things.

## Help Commands

`C-h C-h` Your basic help command; gives you a bunch of options and is pretty much self-explanatory.

`C-h m` Use it to get mode-specific information.

`C-h t` Activates the Emacs tutorial.

`M-x apropos`
: Prompts you for a topic and delivers a list of commands including short descriptions of their functions on that topic. One of the most effective ways of getting help fast.

`M-x manual-entry`
: This is equivant to Unix `man` command but nicer, because the manual entry is placed in a buffer which obeys Edwin commands.

### Other Commands

`M-x mail` Used to send (anonymous) mail.

`M-x info` Activates the info tree; see Chapter 2 for more information.

`M-x load-problem-set`
: Loads a problem set (the current one by default) into Scheme.

`M-x checkpoint-floppy`
: Saves all the files in the '`~u6001/work`' directory onto the floppy in disk drive.

`M-x logout`
: The proper way to logout.

`M-x repl` If a '`*scheme*`' buffer exists, switches you there; else initiates a new read-eval-print loop (creates another '`*scheme*`' buffer). Useful when students wedge themselves by either killing their '`*scheme*`' buffer or saving it under a different name.

`M-x scheme-mode`
: Puts the current buffer into scheme mode (this enables `M-o`, pretty-printing, etc.). All '`.scm`' buffers are supposed to be in scheme mode by default, but sometimes something funny happens. If `M-o` isn't working or things are not getting indented properly, try this command.

`C-c C-c` Cancels the command; gets you out of whatever you were doing.

## 3.3 Bare Scheme

There really isn't much you need to know about the underlying bare Scheme. Sometimes the Edwin xterm window can be used to recover hung up Scheme processes, but that rarely works so don't count on it. Basically, typing `C-g` or `C-c C-c` inside Edwin xterm transfers control from the Edwin window to the Edwin xterm (you can then type in Scheme expressions at the REPL prompt and evaluate them simply by pressing `RET`). Type in `(edwin) RET` to return control back to Edwin. You can familiarize yourself with some other interrupt commands inside the Edwin xterm window by de-iconizing it and typing `C-c` followed by `?`. See the chapter on crises for info on how these can be used to try to unwedge people's Schemes.

Beware that, since the window is readily available for people to play with, students will sometimes wedge themselves by doing just that. In particular, **closing the Edwin xterm window will kill your Scheme process and log you out** (this is probably the only way to *really* make yourself lose by not logging out properly—aside from totally wedging your Scheme—since `C-x C-c` is disabled in the student environment, and even closing your Edwin window is possible only after all the files you care to save have been saved, so they can at least be recovered.). Whenever you see students fiddling with the Edwin xterm, tell them not to.

## 3.4 UNIX/DOS

This is the good old losing UNIX we all know and love. Use your xterm to talk to it (you can also open a shell with `M-x shell` and use that, but most people don't enjoy the experience).

### 3.4.1 The u6001 Home Directory

Most of the relevant files/subdirectories are located in the u6001 home directory. One such file is the '`motd`' file discussed earlier. The file is writeable by hacker but not by u6001, so you have to either be logged in as hacker or "su" to be able to make changes to it. Refer to the section on '`motd`' in an earlier chapter for more information on when and how the '`motd`' file should be changed.

The '`~u6001/psets`' directory contains all the problem sets. The file '`probsets.scm`', again writeable by hacker, is also there; you will need to edit it in order to change the default of `M-x load-problem-set`. See the relevant section in Chapter 2 for more information.

The '`work`' subdirectory is where the student's files are copied from his/her floppy if (s)he logs in with the floppy option. Needless to say, this directory is different on each machine, and students cannot access Fnord! each others' '`work`' directories, for security reasons. There are also three other subdirectories, '`work.~1~`', '`work.~2~`', and '`work.~3~`', used to store the work of the people who were the previous (most recent) users of the machine. This is a safety measure; it allows us to recover students' work when their Schemes get wedged, their floppies get corrupted, and other kinds of natural disasters happen. '`Work.~1~`' contains the work of the most recent user, and so on. These directories get shuffled on student login and logout—see the section below about what happens on login/logout for the exact procedure. Again, the files contained in these directories are not readable or writeable by students because of security issues, so you have to have hacker privileges to be able to recover their work.

### 3.4.2 Telnet and Ftp

You can telnet out of the lab by typing `telnet` *destination* from an xterm. To telnet to Athena, for example, type `telnet athena.dialup` or `telnet athena-x` if you want X applications. When prompted for the display you're using, type the name of the machine you are on—e.g., `sicp-23`. Telnetting *into* the lab is disabled, since we don't want strangers to use the most powerful machines on campus at the expense of 6.001 students.

Similarly, you can use ftp to transfer files to and from Athena or any other accounts you might have, by typing `ftp` *destination*.

### 3.4.3 "Su"ing to Hacker

To "su" to hacker when logged in as u6001, type `su hacker` at an xterm prompt. You will be prompted for a password. Be careful when typing it in; make sure the students don't find out what it is. Once the password is verified, you can do anything hacker can do (your Edwin window will still think you're u6001, but that shouldn't be much of a problem, as you probably won't be using it anyway). When you are done doing whatever it was you needed to "su" to hacker for, type `exit`. Don't forget to do this; if a student discovers (s)he has hacker privileges, a lot of damage can be done to the system.

### 3.4.4 Making Software Changes on Both Servers

There are 46 snakes in the lab, sicp-02 through sicp-47, not including the two servers, sicp-00 and sicp-01, and two printers, sicp-48 and sicp-49. That's 23 * 2, or 23 machines per server.[1] Machines sicp-02 through sicp-24 are connected to the server sicp-00 and the printer sicp-48, the rest to the server sicp-01 and the printer sicp-49. The most important consequence of having two servers is that **all the software changes should be made on both servers**, so that *all* the machines remain up-to-date. Synchronicity is your friend. You should remember that, specifically, when

[1] Whadya know? By the way, the Law of Fives also applies: go ahead and count the number of chapters within this very document and the number of subsections within this particular section.

installing or making changes to the problem set code, editing the message of the day, changing the default of `M-x load-problem-set`, and altering other things in the entire system.

You have to be logged in as/"su"d to hacker while making the changes. If you are making the change from one of the sicp-00 machines—you are logged in from sicp-05, for example—changing the appropriate file will bring all the machines on that side up-to-date (i.e., the change will be made to the code on sicp-00, the server). To make the change on sicp-01 as well, type `cp` *filename* `/nfs/sicp-01/users/u6001/`*filename* at the xterm prompt. For example, to copy the '`motd`' file when in the u6001 home directory, type `cp motd /nfs/sicp-01/users/u6001/motd`. To make sure the change has been made correctly, do `more /nfs/sicp-01/users/u6001/motd`.

### 3.4.5 DOS

The floppy disks use an MS-DOS file format. This is nice, because students can work on their problem sets without having to come in to the lab—i.e., they can type the code in somewhere else, like an Athena workstation Fnord! or a PC, and use their lab time for debugging purposes only. If you know enough about DOS commands to use them to copy stuff to disk, etc. and want to fool around with them, more power to you—just make sure you don't make anyone lose in the process[2]. Usually, though, the only command you need is `M-x checkpoint-floppy` (this is an Edwin command, issued from your Edwin window), used to copy everything in your '`work`' directory to the floppy. The only other thing I've found pretty useful so far that students often ask about is how to see the contents of your floppy disk: in an xterm, type `dosls /dev/rfd:/`.

While we are on the topic of saving files to disk: by the end of the term, checkpointing a typical floppy will prove to be quite a lengthy affair, since students will accumulate all their work from day one on their floppy disk. Please advise them that at least all the '`.bak`' (backup) and '`.asv`' (autosave) files can be deleted from the disk—this will speed things up quite a bit. Specifically, you can do the following:

[2] You can find out more about `dosls`, `doscp(1)`, and so on by typing `M-x manual-entry RET doscp(1) RET`, for instance. The floppy device is `/dev/rfd`, so, for example, to copy the file '`foo.scm`' to a floppy, you would use `doscp foo.scm /dev/rfd:/foo.scm`

1. % cd work
2. % rm *.bak *.asv (this removes files from the 'work' directory)
3. After you do this, when executing `M-x checkpoint-floppy` or `M-x logout`, you will be asked whether these files should be erased from the floppy. Answer yes.

## 3.5 What Happens on Login/Logout

The login/logout procedure outlined in this section has been changed around a lot ever since the system was setup. It is still unclear if it is the right thing. While the description below is accurate as of the time of this printing, it may change over time. The relevant files to consult for reference are '`~u6001/.xsession`' and most of the files in the '`~u6001/bin`' directory, such as '`login`', '`logout`', '`rotate`', etc.

When you are a u6001 user, on logout (this is a clean logout, caused by executing `M-x logout`) the files in your '`work`' directory get shifted over to '`work.~1~`'. The files in the other '`work`' directories get rotated accordingly. The events are, in this order:

1. The '`work.~3~`' directory is deleted.
2. '`Work.~2~`' is renamed '`work.~3~`'.
3. '`Work.~1~`' is renamed '`work.~2~`'.
4. '`Work`' is renamed '`work.~1~`'; protections are changed to make the new '`work.~1~`' directory unreadable by u6001.
5. The new '`work`' directory is created, with appropriate protections.

It is unclear, however, what happens on an unclean logout—e.g., a Scheme crash—since the logout script may never have been run.

Therefore, there exists a mechanism run at login time which should detect unclean logouts, explain the situation, and ask if the user wants to recover any possibly unsaved work. If (s)he says yes, the script prompts for the hacker password. If the student answers no, the '`work`' directories are rotated as usual. See chapter 4 (the section on software crises, case 3) for details.

If someone somehow gets their files rotated away from them, there is a setuid script called '`unrotate`' in the '`bin`' subdir, that prompts for the hacker password (i.e., you needn't be logged in as hacker). To run it, type `~u6001/bin/unrotate` in an xterm. This will do the obvious thing: move the `work.~1~` files up to '`work`', and so on.

When you login as hacker, no rotations happen. Thus, logging in as hacker may be useful when recovering students' work, since you know for sure that it is still somewhere around—in the '`work`' directory if no rotation has been done, or in the '`work.~1~`' directory in case of proper logout. See the chapter on crises for more information on when to login as hacker and when as u6001, and on how to recover students' files.

## 3.6 Hacker Account

When you login as hacker, a Scheme process is not started for you automagically. If you want to start one up, open an xterm and type `edwin &` for a regular Edwin window or `edwin -band 6001.com &` for a student band (probably the only difference you'll notice is that you don't get all the silliness about the floppies, motd, etc. when using the regular band).

The '`hacker`' Fnord! home directory doesn't contain anything really interesting.

Be careful if and when you create temporary subdirectories in the '`hacker`', '`u6001`' or any other directory (sometimes you might want to do that in the process of recovering some student's files, for example). If the files contain anything resembling problem set solutions (and they usually do), make sure they are not readable by u6001 so as to not promote cheating. Try to avoid creating such subdirectories at all; there are usually ways around it. Similarly, if you are using the '`hacker`' directory to temporarily store your own problem set solutions, make sure they are not available for public use. And **please clean up after yourself.** Store your work on your floppy, or ftp it to your Athena account if you have to, but don't leave it hanging around.

There are really no advantages to logging in as hacker in non-crisis situations. (In times of crisis, on the other hand, it may prove necessary to do so, since logging in as hacker does not rotate the '`work`' directories, whereas a normal u6001 might. See the next chapter for more information on these kinds of situations.) When the lab is not busy and you find yourself spending most of your time at the terminal, however, you may use whichever account you find more agreeable. For one thing, the default size of hacker xterms is much more Fnord! sensible than the one used by u6001.

# 4 Crises, Big and Small

Most of the crises that occur in the lab, such as students wedging their Scheme processes or paper jams in the printer, are of a rather routine nature. You will get the hang of these rather quickly and will usually have no problems dealing with them. If something extraordinary happens, however—a server dies, for instance, or it looks like someone is about to immanentize the Eschaton[1]—and you don't know what to do or need help in dealing with the situation (when the server dies, for example, you will probably appreciate a few warm bodies who know the hacker password and can recover people's work present in the lab), use all the means known to you to summon help. Send out e-mail, if you can, since quite a few people on the staff are addicted enough that they will actually read it within minutes after you send it. Don't hesitate to call the head TA, head LA or lecturers at their offices or at home, if needed (you can probably find out their phone numbers by fingering them). Get the instrument desk people to help you out.

## 4.1 Hardware

Whenever you encounter a hardware problem (such as a machine not operating correctly) that you cannot fix yourself, do *all* of the following: report the problem to the instrument desk, send mail to `6001-feedback`, and put a blue tag on the machine, filling in the problem, your name, and the date. The tags can usually be found either on top or in the top drawer of the filing cabinet.

### 4.1.1 One Machine Dies/Acts Funny

#### 4.1.1.1 Circuit Breaker Problems

If a snake is showing no signs of life (specifically, the heartbeat light, which is the first one on the left, is not blinking), the problem is usually a tripped circuit breaker. Unfortunately, this happens more often than it should, because the cheap circuit breakers in the lab are flakey and unreliable. There are two circuit breaker boxes on the 001 side of the lab, one, to the right of the Lambda Lounge, by the help Q board (#1), another to the left of the lounge, close to the blackboards at the back of the room (#2). To recover a machine in case its circuit breaker trips:

1. Get under the table on which the machine is standing. Find one of the cables that lead from the machine to the power outlet, and read off the number written on the outlet corresponding

[1] In other words, bring the end of the world closer.

to the cable (be sure to get the correct cable—you don't want to get the wrong number and make a student on a neighboring machine lose by playing with the wrong circuit breaker).

2. Proceed to the appropriate circuit breaker box, according to the number you read off. For example, 2-23 means it's box #2. Open it and look for the breaker labeled 23. If the number is correct, the corresponding breaker should look different from the other ones: it will not be lined up with the others but will usually be in the dead center position.[2] To recover it, first push it all the way OFF (i.e., opposite the direction the other circuit breakers point in), and only then bring it back to line up with the rest. The machine should now make some noise and show signs of life. It will reboot and come around in a minute.

If you notice that a particular machine experiences the circuit breaker problem on a regular basis, report the problem in a manner described above.

#### 4.1.1.2 Rebooting a Snake

If you cannot get a snake to respond (i.e., the screen is locked up so nothing you type is being acknowledged and the mouse does not respond), you need to reboot it. For soft reboot, press `C-SHIFT-RESET`; it should only take a few seconds for the machine to come around. To powercycle the machine, press the white power button, located at the bottom of the machine, below the disk drive; hard reboot takes a little longer, maybe a couple of minutes.

If the machine does not reboot properly, leave it alone and report the problem (make sure to put a tag on it).

#### 4.1.1.3 A Machine Is Using a Wrong Version of a PS

If one machine seems to be using the wrong version of a problem set or any other file, such as '`motd`', chances are all the other machines on that side of the room—i.e., connected to the same server—are experiencing the same problem. Check this by looking at the file in question on both sicp-00 and sicp-01 and comparing the two versions; if that is indeed the case, and **only if you are absolutely sure which of the versions is correct**, copy the correct version to the losing server (see the section on how to make software changes to both servers in the previous chapter). If you are

[2] Some of the flakier circuit breakers might not change their appearance when tripped. Go ahead and turn the breaker on in the manner described below, but only after making sure it is indeed the right one—i.e., if there is one next to it that appears tripped, maybe you got the wrong number in step 1.

not sure what the correct version is, don't try to guess; simply report the problem to the head TA/LA. In any case, the head TA should be informed if you encounter two different versions of a file.

If the problem seems to be something else and you cannot figure out how to fix it, ask the student to switch to another machine and put a sign on the machine asking people not to use it. In addition, report the problem by following the procedure described above.

### 4.1.2 A Server Dies

Speaking of immanentizing the Eschaton... Murphy's law says that the first server crash must occur the night before a problem set is due, in which case you will notice it momentarily, for it will be accompanied by the sound of 46 students screaming, sighing, taking a deep breath, and doing whatever else people do in times of crisis. Don't panic. Try to calm the students down, too. A server crash is a serious problem, but their work can usually still be recovered. Call for help right away (you will need it even if you can fix the problem itself, to recover people's work as fast as possible), get into the server closet (it is located in the front of the lab, behind the printers), and start investigating.

One of the ways a server can crash is the same way any machine does, by means of a circuit breaker tripping. The consequence of this, as you might imagine, is that all the machines connected to that server panic and crash as well. To fix the problem, do what you would normally do in a case of circuit breaker blowing (consult the section on circuit breakers above). The circuit breaker box for the servers is located inside the server closet.

If the server crashes for an unknown reason, and is *completely catatonic*, the only thing you should try is powercycling it (see the section on rebooting above): turn it off, wait a few seconds, turn it back on. If this doesn't help, do not spend the time in an attempt to figure out what the problem is; get help immediately. Do not even do that much (i.e., don't powercycle the server) if it seems to be running, since that is usually an indication of a different problem. Powercycling a system with an active filesystem may cause data lossage, and we only want that as our last alternative.

It will take quite a while for the machines to come back to life. When they do, start recovering students' work (see the software section below to find out how). Be extremely careful. If you can find a student who did `M-x checkpoint-floppy` recently and thus doesn't care if his/her files get recovered, make his/her machine your first guinea pig. If and when the reinforcement arrives, the total recovery shouldn't take long.

### 4.1.3 Printer Problems

If you think there is a paper jam in one of the printers, lift the cover of the printer by pressing the button on the left of the cover, and look inside. If no paper is visible, try pulling out the paper trays and see if any paper is stuck on the bottom. In addition, there is a side access door; open it by pressing the left side of the printer in the right magic place–the access door should pop open. In any case, you will usually be able to pull the paper out somehow. If there is no paper seen anywhere inside the printer, maybe the paper jam is not the problem.

When the printer complains that the toner is low and does not print:

- If the number following "toner low" is 1 or 2, you don't need to replace the toner cartridge. Just lift off the cover, pull it out, shake it up a little, and put it back in. The printer should come back to life.
- If the number is 3, the toner cartridge needs to be replaced. Go to the instrument desk and ask for help.

If a printer goes catatonic with some weird error message, you can try powercycling it.

If one of the printers dies mysteriously or experiences a problem you cannot fix yourself, report the problem. Ask the students on the side of the room with the broken printer as their default to switch printers, by executing `M-x set-variable RET lpr-switches RET '("-Psicp-48") [or 49]` in Edwin. If both printers die, we are in trouble; get help fast.

## 4.2 Software

One of the more common software problems is the lack of synchronicity between the code on the two servers—consult the section on making software changes on both servers in Chapter 3 Fnord! to find out how to bring the wrong versions of the code up to date. The two others are stray Scheme processes and people with wedged Schemes.

### 4.2.1 Machine is Slow: Killing Extra Scheme Processes

If a student complains that his/her machine is slow, check the network lights. If the machine is swapping, that is usually a sign that there are some extra Scheme processes running on it, left over from previous users (often the machines will be thrashing to death and students won't even notice it). Here is how to look for/get rid of them:

1. Ask the student to save, Just In Case.
2. Open an xterm.
3. % ps -ef | fgrep scheme
4. If there's more than one "scheme -edwin" process, look for the one that has the "xterm ... edwin ..." process as a parent (the lines in the fgrep output should have matching magic PID's).
5. Kill all the *other* scheme processes (using kill without options—i.e., % kill *PID*—works fine if you're doing it from a student's xterm).
6. % exit

You can also get rid of stale Scheme processes by typing `~u6001/bin/killschemes` at the prompt; this will run the appropriate script. However, doing so will kill *all* the Scheme processes currently running on the machine which have u6001 as their owner—**including the student's Scheme**. Needless to say, you should not do this when logged in as u6001, unless you desire to kill your current Scheme process and be logged out without even having a chance to save your work on disk.

If the machine is slow but no extra Schemes are running, it may be the case that the student is hosing it down by writing losing Scheme procedures (for example, creating a lot of streams will slow down a machine considerably). Tell the students to at least kill their xclocks, xloads, xterms, etc. to speed it up a little. If the problem seems to be something else, report it: it may be a hardware problem.

### 4.2.2 Unwedging People With Dead Schemes

When a student's Scheme crashes, you get to (joy!) recover his/her files. Naturally, you can only recover as much as they have saved, so if the student doesn't Fnord! believe in `C-x C-s`, some of his/her work is bound to be lost. Students should get into the habit of saving *often*; beat it into their heads every time a crash occurs and they call you to recover their work. The best way to save is by doing `M-x checkpoint-floppy` after writing each significant chunk of code, and especially *before* running the code that is likely to crash the machine (this happens a lot when dealing with streams, for example, because it is easy to get into an infinite loop and run out of memory)—this way, your help may not even be needed, since the versions of the files read in from the floppy on login will be recent. If `M-x checkpoint-floppy` seems as too much trouble, the students should at least do `C-x C-s` as often as possible.

There *is* a fallback, though. The system creates and updates an '`.asv`' (autosave) file for each of the '`.scm`' files in the '`work`' directory. These '`.asv`' files prove to be pretty useful; they get updated often enough that most of the time they contain the more recent versions of the files than the '`.scm`' files themselves. Don't forget about them—sometimes they will save a student a headache by recovering a good chunk of work that is otherwise lost. After recovering someone's files, show him/her the `.asv` versions and ask him/her if those are more recent. If that is the case, and if the student would like the contents of `.asv` files to be copied to the '`.scm`' files, from an xterm,

1. % cd work (if you're not already there)
2. % cp *filename1*.asv *filename1*.scm
3. % cp *filename2*.asv *filename2*.scm ; and so on

With regards to the machine crash itself, there are several kinds of lossages which require recovery of students' files, and each dictates a different kind of treatment. We discuss them all below. Note that this is just *one* way of doing things; there are other ways to go about recovering files, some riskier than others. You may like something else better—use it at your will, but please make sure it is safe as to not make students lose. Here, I tried to come up with procedures which are both fast and safe.

## Case 1: Edwin xterm Saves the Day.

If the Scheme is hung up, but you can still manipulate your mouse, it is possible (although not very likely) that you can bring the Scheme back to life with the help of the Edwin xterm window. After de-iconizing it, (if not already in REPL) press `C-c C-c`; at the prompt, type `(edwin) RET`. If this doesn't liven up your Scheme, there's not much you can do to recover the process itself (option '`R`', "hard reset possibly killing Scheme in the process," though it looks vaguely promising, seems to justify the warning every time, indeed killing Scheme in the process).

Try at least recovering the files by typing `(save-editor-files) RET`. This will save all your buffers the files for which already exist; it will also ask you whether you want to save other buffers, such as '`*scheme*`', '`*transcript*`', and so on. That can prove useful to students who would like to save the transcript of their work—to figure out what went wrong, for example, or in order to try to get the bug to repeat. `(Save-editor-files)` is an excellent command (when it works), because the students don't lose any work. However, it doesn't solve the problem of writing the saved files out to the floppy. You still have to follow the procedure outlined in the next section to make sure the files which are now in the '`work`' directory find their way to the student's disk.

## Case 2: You Can Open an xterm.

If you can get no help from Edwin xterm, but you still have control of the screen—i.e., you can open an xterm—your best bet is to do the following (the last eight steps of this procedure, 5 through 12, may also be used to recover the work of a previous user of the machine who has lost his/her floppy or some such thing; the only change is in step 8, where the appropriate directory—'`work.~2~`' or '`work.~3~`'—should be substituted for '`work.~1~`'.):

1. Open an xterm.
2. % ps -ef | fgrep scheme
3. Find the PID of your scheme -edwin process.
4. % kill *PID* (this logs you out)
5. Now log back in as u6001 (not hacker!), with the disk in drive.
6. Open another xterm.
7. % su hacker
8. % cd ../work.~1~
9. % mv * ../work
10. % chown u6001 * (important!)
11. % exit
12. In Edwin, do `M-x checkpoint-floppy`, so the student can go crash again as soon as (s)he re-evaluates the code, and not really care.

The advantage of the above method is that it minimizes disk accesses, and leaves the machine in a state where the student can continue work as soon as file recovery is complete, without having to logout. Logging in as hacker in this kind of situation is unnecessary and will only complicate things. Since you issue the `kill` command, the logout should proceed as normal, in the sense that the '`work`' directories will be rotated. Thus, you will be able to find the student's files in '`work.~1~`'.

## Case 3: No Keys are Responding

If the screen is locked up completely and you can't open an xterm or even move your mouse, your only alternative is to reboot the machine and log back in to recover files. It should still be safe to login as u6001. Presumably, if the logout was clean (i.e., if the directories got rotated correctly), you can follow the procedure outlined in case 2 above (steps 5 through 12). If, on the other hand, the logout script did not get run, this will be detected on login.

As mentioned earlier, the login script checks for unclean logouts and, if one is found, explains the situation to the user. If the student wishes to recover files, the system then prompts for the hacker password. With the correct password entered, directories are left unrotated, and the student can continue the login process by selecting the **N option (practice login), not the default (L) option**. Note that this means the student should use `M-x checkpoint-floppy` before logging out, since `M-x logout` won't try to save anything to disk. Besides that, however, everything should appear just the way it was before the crash—in particular, the student's files will still be in the '`work`' directory, and nothing will need to be moved.

## Case 4: Student's Work is in '`work.~3~`'

While logging in as hacker may be considered a safe thing to do in some situations, it is a *necessary* action if the student's work is *already* in the '`work.~3~`' directory, since logging in as u6001 will hose it all. Here is the entire procedure, remarkably similar to the one above (the major difference is that you need to create a directory to temporarily store the student's files):

1. Login as hacker.
2. Open an xterm.
3. % cd ../u6001
4. % mkdir joe (for example; the name of the directory you create should not be something obvious like '`tmp`' or '`temp`', in case another LA is doing something like this at the same time; that's why I use the student's name)
5. % cd work.~3~
6. % cp * ../joe
7. Logout; log back in as u6001.
8. Open an xterm.
9. % su hacker
10. % cd ../joe
11. % mv * ../work
12. % cd
13. % rmdir joe (clean up after yourself!)
14. % exit
15. In Edwin, do `M-x checkpoint-floppy`.

Don't forget to remind students that '`.asv`' files may be more up-to-date and that they may recover some of their lost work that way.

# 5 How to Deal with Students

Back in the old days, when LAs were called Lab TAs, when all we had were ones and zeros, when Atlantis ruled the Earth... anyway, the point is that missing T in LA, the one that stood for Teaching. Remember what your function in the lab is? Fnord! Helping students, dealing with their problems and answering their questions, however easy/hard/stupid/ridiculous they may seem (we all know by now Fnord! that ALL questions are good questions), trying to *make them understand*.

## 5.1 Helping a Student

When you are helping students, don't do their work for them. Ask them to show you the different pieces of code and to try running it for you. **Do not take over at the keyboard** for them. Although this may be frustrating when the student is slow, that is the only way they will learn. When the students are off on the wrong track, try to guide them onto the right track, but don't solve the problem for them. Don't let students badger you into telling them the answers—that's not your job.

Try not to spend a lot of time helping a single student, especially if the lab is busy. Point the student in the right direction and let him/her think about it some more.

## 5.2 Collaboration Policy

It is a very good idea to get students to help each other, especially when the lab is crowded and the LAs are swamped. 6.001 collaboration policy states that "we are very much in favor of collaborative work: students working together in small groups on problem sets—in working through problems, planning programs, and programming in the lab—so long as everyone is involved in all aspects of the work." Take advantage of this. For example, when you encounter a person with a question or problem similar to the one you just answered for someone else, ask the person you just helped if (s)he would mind helping another student. You will find that most people are quite cooperative in this respect. Besides lessening your workload, this technique actually aids the students, who may understand the issue better after explaining it to someone else.

## 5.3 Do not Get Angry

Do not get irritated at students, or, at any rate, don't show it when you do.[1] Helping them is not always easy and can be frustrating. When you think you are about to lose your cool, don't take it out on the student—that will only make the situation worse. Take a step back and try another approach, or if there is another LA on duty, see if (s)he can help.

In general, it helps to be relaxed and armed with a sense of humor. Instead of blowing up when you are getting frustrated, laugh.

## 5.4 What to do When you Feel Like you're in a Bad TV Show

Here is the (paraphrased) text of an e-mail message sent to `6001-feedback` by a student (we'll call him Bruce) after one of the 6.001 quizzes:

> [lecturers],
>
> i came to mit because i wanted to graduate from a decent college and get a reasonable job. your test is going to stop all that. i can't believe you did this to me. what am i supposed to do now? i can't take any more. i hope you are below me. this is on your heads.
>
> goodby.

This is the kind of message you should not dismiss as an ordinary flame, as we (a couple of us LAs on duty) once did when "Bruce" came in to the lab to write it. As it turned out, this particular bruce *was* just flaming; some students, however, may be quite serious in their suicidal tendencies. If you encounter anything of the sort, start acting as if you were a character in Beverly Hills 90210: try to find out how serious the student is and, if you think there is even the slightest chance that (s)he *is* serious, call the lecturers, the CPs and/or anyone else you think could help or should know about the situation. Please take this seriously; MIT is a stressful place, and 6.001 can be a stressful subject, and things like this do happen.

---

[1] I guess, the best analogy is "The customer is always right."

## 5.5 Quick Questions

*All* questions are good questions; however, *no* question is a quick question. On the slowest and on the busiest nights, at the end of the term as well as at the beginning, invariably, you will have students coming up to you saying: "Can I ask you a QUICK QUESTION?" You should learn that *that* in itself is *the only* quick question anyone ever asks. There are no others. This is ok if the lab is not busy and noone is waiting on the queue. Even when there are a couple of people waiting for help, you might try to answer the question on the fly—if it is truly quick, you will not need to go to the student's terminal (if, on the other hand, (s)he starts by "I wrote this piece of code, and it doesn't work," throw up your hands in disgust and tell him/her to enqueue him/herself and wait for his/her turn just like everyone else). The situation does not improve on the busier nights. In short, pretty soon you find that the only workable answer when someone says "I have a quick question" is "I have a quick answer... NO[2]" Just make sure you have a nice friendly grin on your face the entire time.

All Hail Discordia! and now even I am finished, and the only quick question I have left is: did Joe Malik ever have any dogs? and if he did, how many? and what happened to them after he went underground? and what did he do with the dogfood? and if he didn't, what were the neighbors complaining about?

The rest will be printed in Heaven.

[2] Or 42, or 23 Skidoo, or anything else you find amusing.

# Concept Index

# Table of Contents