Computer Science Club

A Student Chapter of the ACM

The ACM Programming Contest Environment
Contestants' Instructions

Ian Goldberg
Michael Van Biesbrouck

July 9, 2007

1 Logging In (X Terminals)

You should see a window in front of you asking for your userid and password. If you pre-registered, you should have received your userid and password by email. If you have not yet been given your userid, find someone to tell you. At some point, you should get a slip of paper with the following information (for example):

```
Hacker College A:
Login: acm042        Password: ctdalek
```

The `acm`*nnn*, where *nnn* is a 3-digit number, is your userid. Enter this userid to the login screen. You may be asked for the password several times, as the contest environment will log you in to the machine where you will do all of your work. If all is well, the background of your screen should change to the CSC logo (see the top of this page), and two windows and a clock should appear.

One of the windows should be asking for a Contest Password. This is your shell window. All your work will be done in this window, or additional windows like it. The other window is the score window. The current standings, clarifications, and judges' responses to your submissions will be in this window.

Use the mouse to move the pointer to the window in which you want to type before typing there. If the pointer is pointing to the background or the clock, keystrokes will be ignored.

At this point, you should enter your contest password (as given on the slip of paper mentioned above). **Be careful! If you enter it incorrectly, you will be logged out and will have to start over.** If the contest has not yet started, a message will appear telling you to wait. **Do not open your problem packages until you are told to begin.**

1.1 Logging In (PCs)

If your team is assigned to a PC, instructions for logging in will be given in the computer lab.

2 The Score Window

Take a few minutes to explore the score window. If you press `h` (when the pointer is in the window, of course), it will display a help screen listing the various keys you can use and their actions.

The score screen (`1`) is a list of the current standings. Your team will appear in boldface. The clarifications screen (`2`) is a list of messages from the judges to everyone. The submissions screen (`3`) is a list of messages to just your team. The most recent messages will be at the top, and if a message comes in for a particular screen, that screen will automatically be displayed.

If a screen does not all fit in the score window, you can use the other listed keys to go up or down by one line at a time, or by pages at a time.

Finally, `q` will remove the score window. **This is probably not what you want to do.** If your score window closes, and you want it back, use the command `watch` from your shell window (once it activates).

3 The Contest Environment

When the contest begins, a message will appear stating that you may begin, and you will be given a prompt like the following:

`acm`*nnn*`[`*x*`]$`

Where `acm`*nnn* is your userid, and *x* will be `1` initially, and increment for every command you type. The commands available to you are as follows:

3.1 `ls`, `cat`, `more`, `vi`, `xedit`, `jove`, `time`, `diff`, `rm`, `cp`, `mv`, `man`, `echo`, `clear`

These commands are used in the same way as their UNIX counterparts. A brief synopsis of each follows:

ls	List the files in the current directory
cat	Display the contents of a file
more	Display a file one page at a time
vi	An editor
xedit	Another editor
jove	Yet another editor (EMACS)
time	See the CPU usage for your program
diff	Show differences between two files
rm	Remove a file
cp	Copy a file
mv	Move (rename) a file
man	Get help on a command or C function
echo	Output a line of text
clear	Clear the screen

`del` and `erase` are synonyms for `rm`. `rename` is a synonym for `mv` (**the command `rename *.c *.C` does not do what DOS-users will expect it to; this can delete important files**). `copy` is a synonym for `cp`. `dir` is a synonym for `ls -l`.

`more` is actually `less`, a nicer program. You can call it `less` if you want to.

`jove` is the suggested editor for new users. You can get by if you know that `CTRL-X CTRL-C` quits and `CTRL-X CTRL-S` saves your current file. Other useful things: `ESC x auto-indent-mode` toggles auto-indention, `ESC G` is `goto-line`, `ESC x describe-bindings` tells you what every key does and `CTRL-X ?` tells you what just one key does. If you don't understand the brief explanation for a command, say `find-file` (invoked by `CTRL-X CTRL-F`), use `ESC x describe-command find-file`. When you are in `ESC x` mode, `?` tells you what you can type and `TAB` types for you if there is only one choice. Some people might want to use `ESC x number-lines-in-window`.

3.2 `gcc`, `pc`, `bpc`, `lpr`, `xterm`

These commands behave like their UNIX counterparts, but take restricted options.

`gcc` is the C and C++ compiler. It can only take the options `-o` *filename* to set the executable's filename, and `-W...` to set various warning levels. It is generally used as `gcc qA.c`. This will compile `qA.c` as a C program and call the resulting program `qA`. The input name must end in `.C` for the program to be compiled as C++. If an output filename is not specified, it defaults to X where the input files was X`.C` or X`.c`. Your code will be compiled as if `gcc` was run with `-O2 -ansi -lm -lcfix`. The `-O2` option turns on most compiler-level optimizations; concentrate on good algorithms during the contest.

The most useful warnings to turn on are `-W` or `-Wall` and `-Wunused`. Type `gcc -W[TAB][TAB]` to get a list of more warnings (the list is not complete, but many warnings are for obscure purposes that do not apply to programming contests). Most contestants will want to turn on some warnings.

`pc` is the Pascal compiler (using `p2c`). It can take the `-o` *filename* option, like `gcc`, above, and also default to X if the input file is X`.p` or X`.P`. It will use the same optimizations as `gcc`. If the file name ends in `.p`, `pc` compiles HP Pascal, which is very similar to Turbo Pascal. Use file names ending in `.P` for Berkeley Pascal.

`lpr` is the program used to print files. It is used as `lpr file1 file2 ...`. This will send `file1` and `file2` to be printed. `print` is a synonym for `lpr`. Printouts will be brought to your station by a proctor.

`xterm` creates a new shell window. It does not take any arguments. Note that you should type `xterm`, not `xterm &`.

`jc` is the Java compiler. It has one parameter: the name of a file containing a Java class definition. The file must be named X`.j` where X is the name of the main class defined in the file. If the compilation is successful, it creates an executable file X which can be run as described above.

`sc` is the Scheme interpreter (mzscheme). It has one parameter: the name of a file containing exectuable Scheme statements. The file must be named X`.s` where X is the name of the main class defined in the file. The scheme commands are run immediately. (Note: this is different from all the other language processors which require that you first compile and then run your program.) Your program may read input from standard input and write to standard output, as described above.

3.3 `watch`, `submit`, `history`, `help`, `exit`

These commands are unique to the contest environment.

`watch` will start a new score window, this should only be necessary if it happens to close.

`submit` takes a filename as an argument, and submits that file to the judges. The filename must end in X`.c` if it is a C program for problem X, X`.C` if it is a C++ program for problem X, X`.p` or X`.P` if it is a Pascal program for problem X, or X`.r` if it is a clarification request for problem X. X will be a letter. For example to request a clarification on problem C, state your request in a file called (for example) `qC.r`, and then `submit qC.r`. Creating files with names such as `mary2_B.c` may be helpful.

Java programs must be named exactly X`.j` where X is the name of the problem.

`history` lists all the command lines you've typed in the current window. If you say `history` n, where n is a number, it will list only the last n command lines.

`help` will give a list of all available commands. The list will be the commands listed here, as well as any executable files in your directory.

`exit` will quit the shell and log you out. **You probably don't want to do this.**

4 Shell Interaction

The shell uses the Perl version of `readline`. This means that you can use arrow keys to edit your current command and move through the command history. EMACS-like key bindings work, but `vi` key bindings are not supported. This means that `CTRL-A` will move to the start of line and `CTRL-D` can be used to delete characters, just like in `jove`. If you are not familiar with EMACS key bindings you will still find command-line editing to be easy since the arrow keys and backspace work just like you would expect them to.

A handy feature of the shell is command-line completion. At any time while editing a command you can press `TAB`. The shell will guess what you might want to type at that point in the command. If there is only one choice, it will type it for you. If there is more than one choice you can press `TAB` again to get a list of choices.

Completion is sensitive to what has already been typed. In particular, it knows which programs you can run, time and pipe to. It also knows a lot about the options to `gcc`, including useful warning options.

This behaviour can be enabled and disabled using the commands `readline` and `noreadline`. By default it is enabled.

5 Other Shell Things

You can use Bourne-shell-type redirection to and from files using `>` and `<` (to redirect error output, use `2>`), as well as piping to `cat` or `more` (for example, to see your (long) output of the program `q1`, when run on the input file `q1.in`, you could use `q1 < q1.in | more`).

You can use limited C-shell-style history substitution. `!!` will repeat the last command line; `!*` are the arguments of the last command line; `!$` is the last word on the last command line; `!`*n* repeats command number *n* (as indicated by `history`); `!-`*n* repeats the command *n* lines ago (so that `!!` is the same as `!-1`); `!`*word* repeats the last command line starting with *word*.

6 Other GUI Things

The three mouse buttons have various functions, depending on what you do with them. If you click on the background (showing the CSC logo) you can do three different things:

left	Commands to manipulate windows (raise/move/delete)
middle	Create calculator, editor, submit, watch and shell windows
right	Adjust mouse speed, keyclick and bells

You can select text with the left mouse button and paste it with the middle mouse button.

The scroll bars use all three mouse buttons. Use the middle button to position the slider. Use the left button to scroll down and the right button to scroll up. The distance that you scroll will depend on how far down the scroll bar you click. At the top it will move only one line, at the bottom it will move an entire page.

If you exit the initial contest shell window, you will be logged out.

7 Security

If the judges catch you trying to break security, you will be disqualified. Breaking security includes running programs that are not intended to be available to contestants, doing anything to any files in another directory, and anything else you shouldn't need to do to write this contest.

This is not a security cracking contest.

8 How to Test Your Programs

This is the standard process for creating and testing a program:

`vi A.c`	Write the program. Note that the problem letter is a capital.
`gcc -Wall -Wunused A.c`	Compile the program. You may want to turn on warnings for C/C++.
`vi A.in`	Create a test input file.
`vi A.diff`	Create a file with the correct output.
`A < A.in > A.out`	Run the program on the test data.
`diff A.diff A.out`	Compare correct and real output. `diff` *should produce no output.*

If you do not test your program this way, then you are not testing it the way that the judges will test it.

9 Pascal Programs

Note that there are two different versions of Pascal available. Both are case sensitive — you must use lower case for all keywords (`begin`, `program`, `integer`, etc.).

Default output formats are wrong for this contest. Use `writeln(x:0)` to print an integer because `writeln(x)` may give you a format error.

Integers are 32 bits (-2^{31} to $2^{32} - 1$).

Strings are not implemented in the traditional Pascal way. You can make them as long as you want, but you must give an explicit size when declaring them. Use `copy()` to truncate strings (assigning numbers to the zeroth position in the string will not work in all cases).

Never use `string()` to turn a character variable into a string. This conversion happens automatically (and correctly), but `string()` is broken.

If the name of a variable or a subroutine is the same as one of the standard C functions, the compiled program may behave unpredictably. Some examples are read, write, and time.

10 Java Programs

Your entire Java program must be contained in one source file.