

# ISG Course Accounts

## *What Do I Do Now?*

This tutorial is intended as a brief reference guide for a tutor who has been assigned to a course that uses electronic assignment submission, and/or generated cover pages for written assignments.

Other tutors might find this of use for determining how to get a list of all students registered in the course, for determining how to maintain the files that constitute a course's web page, answering email directed to the course account, and publishing student marks in a secure fashion.

# Agenda

- **Structure of Course Account**
- **Maintenance of Course Account**
- **Assignments**
- **Email**
- **Giving Access**
- **Online Marks Database**

Each course account follows a basic structure. For maintenance purposes, it is important that you do not change this structure.

There are several important files and directory structures needed for handling student assignments. Those will be discussed in detail, as will other useful features such as providing access to the course account for instructors, and publishing student marks through the online marks database.

## How Do I Get In?

- Use `rlogin` or `rsh` or `ssh` i.e.,

```
rsh localhost -l csNNN
rlogin <undergrad-machine> -l csNNN
ssh csNNN@<undergrad-machine>
```

**where NNN is a course number**

You should have already been given access to the course account. If not, contact your ISC. You will first need to login to your UNIX account on one of the `student.cs` machines. The course accounts do not have passwords for security reasons, so all access is through a utility such as `rsh`, `rlogin`, or `ssh`, all of which have manual pages available (see `man rsh`, `man rlogin`, `man ssh`).

You can use a variety of ways to connect, but `ssh` is preferred since it is more secure. Note that if you are already logged into one of the `student.cs` machines, you can just use `localhost` as your hostname and it will log you into the same machine.

It is recommended that you use one of the CS CPU servers to ensure that you are working with the latest software versions installed. There are no guarantees that you will have the correct versions available if you use other machines. You can determine the current list of CPU servers via the command:

```
hostselect attr=cpu +listall
```

if you are already logged into a CS undergraduate machine.

# Important Directories



- **archives**: holds previous term's information
  - web page
  - assignment information (text, provided files, marking scheme, solution set, scripts, errata page)
  - exams (text of midterm and final exam, solutions, and marking schemes)
  - marks
- **bin**: useful scripts written by past tutors
- **cifs\_exports**: (rarely used) used to make available files/libraries automatically for students on NEXUS machines
- **course**: contains assignment solution sets, marking schemes, files used by `submit`
- **handin**: `submit` stores student submissions here as well as some files used to determine late submission; each assignment subdirectory has a file that controls what is expected as part of the submission
- **marking**: scripts for the automated compilation and testing of student submissions; may attempt to generate marks as part of the test procedure
- **pub**: contains HTML for course web page and files available to students
- **public\_html**: symbolic link to `pub` directory
- **u**: all users with access to the course account have a sub-directory here

## Class List

- `.classlist`
- Read-only, updated regularly by CFCF from Registrar's info
- Do **not** change permissions on this file; instead, use:
  - `.stafflist`
  - `.exceptions`

The `.classlist` file is a read-only file that contains a list of all students currently registered in the course. This list is built by CFCF from data imported by the Registrar's office. It is updated on a regular basis, so it is important that you do not change it since any changes will be overwritten.

If you need to add a student temporarily so that they use the course's web scripts and submission mechanisms, they should be added to the file `.exceptions` in the following format:

```
99999999:userid
```

Course tutors or instructors who wish to use the course's web scripts and submission mechanisms are placed in the file `.stafflist`.

Note that these are only used by Byron Weber Becker's CGI web scripts as part of an authentication mechanism, so if your course has switched to using UWdir/QUEST passwords as an authentication mechanism, you don't need these files.

## Web Site

- `public_html`
- **Ensure list of course staff is up-to-date at beginning of term**
- **Update announcements and marks regularly**

The default location for all web pages is in the directory `public_html`. It is common to have a sub-directory for term specific information, and to have all other common information within `public_html`.

Assignments, provided files/documents, exam information, and grades will all be made available from this central location.

Each course should have a web page devoted to important announcements, even if the course has a newsgroup.

It is a good idea at the start of the term to update the list of course personnel, and ensure that none of the links anywhere in the web space are broken.

## Handin Directory

- **Student submissions, divided by assignment and then userid**
  - `.subfiles` list of files to submit for an assignment
  - `.lates` list of students who handed assignment in late; maintained by scripts `deadline`, `submit`, and `unlate`

If you are using electronic assignment submission via `submit`, first check that `/u/csNNN/.submitrc` exists and that you know how it has defined the variables that control `submit`'s behaviour. This includes whether or not file filtering has been turned on or not. If it has, the definition of `file_filter` specifies the name of the script applied to each file. If the script produces output, the file is rejected. (This is useful for rejecting provided files that should not be changed, MS Word documents, files over a size, etc.)

Every student submission is copied into the `handin` directory, under whatever name was specified as the assignment name (the same name is used by the `submit` command, so if you call the assignment `A0`, you must "`submit csNNN A0 .`". Each assignment directory contains a file `.subfiles` that lists the expected files for this assignment. Simple wildcards can be used, such as `*.java`, but not `*.{h,cc}`.

If late assignments are allowed, the `.lates` directory will contain an empty file for each student who used a late, named with the student's userid. This information is used by the `submit`, `deadline`, and `unlate` scripts. `deadline` is used to specify the end of ontime submission, and the end of late submission, while `unlate` returns a late to a student (useful only if students are allowed a limited number of lates in the course).

# Scripts

<http://www.student.cs.uwaterloo.ca/~isg/scripts/>

- **Purposes:**
  - **Cover page**
  - **Electronic submission of assignments**
  - **Test compilation**
  - **Auto-marking**
  - **Batch testing of assignments**
  - **Printing of assignments**

A number of scripts to assist in common tasks have been written. They and their documentation are available at the listed URL above.

Note that the scripts don't care if the user is registered in the course or not. Most notably, a student not registered in the course can submit at any time, and if the course has a lab fee, a student cannot submit who is registered in the course and hasn't paid the lab fee (actually, there's a small time period when they can, to allow for late registrations in the course, but it's easier to just say they have to pay the lab fee as soon as they register).

## Cover Pages (Written Assignments)

- Configuration file `.coverrc`
- Command `makeCover`
- For each assignment, add list of questions and marks per question to `.coverrc`

The file `.coverrc` specifies which assignments have had a cover page defined so far, the number of questions per assignment, and the amount of marks associated with each question. Since the command `makeCover` uses this to generate a postscript file, there's not much leeway in the formatting of the cover page unless you create your own version of `makeCover`. For example, CS246 created a customized version that allowed a line of text to be added to the top of the cover page that listed all members in the project group, and a blurb such as "The TA can take up to 5 marks off for spelling and grammar." An example of the CS246 `makeCover` and `.coverrc` is available on the ISG web site.

There is a web interface to `makeCover` at

<http://www.student.cs.uwaterloo.ca/~isg/makeCover>

`makeCover` script is documented on the ISG web site.

# Electronic Submission

- **Configuration files:**
  - `.submitrc`
  - `handin/<Assignment>/.subfiles`
  - `bin/file_filter` (see `.submitrc`)
- **See `man submit`**
- **Some courses have a web-form that allows students to check their submissions**

`.submitrc` specifies the number of lates, if any, whether a new submission completely replaces the old submission or is added to the previous submission, the location of all files created/used by `submit`, the location of the student electronic submissions, and whether or not file filtering is performed.

As previously mentioned, the `handin` directory's `.subfiles` lists all expected files, while the file filtering script is applied to each file and rejects the file if the script produces output.

There are manual pages online for each of these. See `man submit`, `man submitrc`, `man subfiles`, and `man file_filter`.

While some courses have web forms that allow students to check on their submissions (and maybe the number of lates used, if applicable), Isaac Morland is working on a web interface to `submit` that allows students to submit their files via the web form.

# ISG Scripts

- **Scripts for testing and printing use**

```
rst <assign> {p|m} <suite>  
'<students>' [<timestamp>]
```

**where:**

- `rst` is `/u/isg/bin/rst`,
- `p` = printing,
- `m` = marking,
- `suite` = name of directory holding scripts

The course account should already have `/u/isg/pub/bin` in its search path. If not, you'll have to modify `.cshrc` / `.bashrc` to include this path.

`rst` is used to run student tests for a particular assignment. If you use the `-p` flag, postscript files are created for printing purposes. If you use the `-m` flag, ASCII text files are created. The `suite` identifies which set of scripts to run. You can either specify a single student name (without quotes), multiple names within quotes, or simple wildcard expressions within double quotes (for example, `"*"` tests all students, while `"[a-d]*"` gets all userids that start with 'a', 'b', 'c', or 'd'). The `timestamp` distinguishes separate runs. Since each run creates a directory in `/tmp` as `/tmp/.csNNN.<assign>.<p/m>.<timestamp>`, you may need to remove that directory as well as the one in your marking directory before attempting to run `rst` again.

`rst` can be used to perform test compilation of student submissions, run them against common data, and assign marks to test results to be recorded in the marksheet when the assignment is printed. `rsta` is used to create a solution set's test results for comparison purposes during the automatic marking as well as test each student submission specified.

More detailed documentation and examples are available on the ISG web site.

# Test Compilation

- **Various files:**
  - `public_html/cgi-bin/requestCompile`
  - `bin/testCompile`
  - `marking/<Assignment>/testm.testcompile`
- **Some courses have a web-form that allows students to test compile their submissions**
- **May also run some simple test cases**

Only a few courses provide a test compilation feature. It is generally used to ensure that a student has submitted all necessary files, and that they compile together in the test environment. If makefiles are part of the assignment, the submission of a valid `makefile` can also be verified.

There is usually a web form that allows the students to submit their requests. The requests are then collated, and a `crontab` job runs `bin/testcompile` regularly to perform the test.

`testcompile` usually has another script, `testcompileLocker`, that ensures multiple test compilations are not running simultaneously since that can cause strange errors and failures in the test. `testcompile` can then issue the appropriate commands for testing the assignment's compilation, or more generically, use `rst` to run a common set of commands for test compilation. For example, in CS343, the students always have to submit a `makefile`, so there is a generic set of test compilation scripts that only need to read a particular assignment's list of `makefile` targets and execute them to perform the necessary work.

A short configuration document is available on the ISG web site.

# Auto-Marking

- **Done in batch after assignment due date, possibly as part of printing**
  - `marking/<Assignment>/testm.auto/`
    - `cases/test*.{in,desc,out,value}`
    - `runTests` script to run tests
  - `marking/<Assignment>testp.auto/`  
`computeMarks` calculates grade to include in marksheet

The auto-marking and batch testing of student submissions is a complicated topic, so you are referred to the ISG web site for more complete documentation as well as sample scripts, though there may be some already available on the course account that you can modify to suit your current purpose.

All scripts are expected by `rst` to be located in the marking directory. There will be a subdirectory for each assignment, and at least one test suite. If the test suite is designed to perform automarking, you will probably want it in a "p" suite so that the results can be included in the assignment marksheet printed for each student submission. If the results are strictly informal feedback emailed to the student, the "m" suite is sufficient.

Each suite generally provides a `runTests` and a `computeMarks` script, though the former may be omitted, depending upon your purposes. If you are performing automated marking, it is a good idea to use as general a script as possible, so that you can put the variances in the test data provided. A common procedure is to provide a `test.in` file that consists of the test input, a `test.desc` file that provides a textual description of the test (useful to include in the marksheet), and a `test.out` file, the expected output. Some courses have also added a `test.value` file that lists the number of marks for a successful pass of the test. You then provide a directory for each test, so your script only needs to loop through each subdirectory of the test directory.

# Batch Testing

- **Similar to auto-testing**
- `marking/<Assignment>/testm.batchtest/`
  - `runTests` script to test submission
  - `computeMarks` script that may do little
  - `data` directory of input files
  - `answers` initially empty directory
  - `solutions` used to compute answers

In order to batch test (and automark) every student submission, you first need to create a solution set's answers for comparison purposes. This requires that `rsta`, a front-end to `rst`, be run initially. This will use the solution set stored in the `solutions` directory, and the `data` directory's input files, to create the files stored in `answers` by using `runTests` to create the expected output.

`rsta` will then test the expected answers against those of the specified students. It is a good idea to do this on a sample of student submissions well before the assignment deadline to test your own solution set and answers since the students may often make assumptions that your scripts aren't prepared to handle (at all, or correctly).

# Assignment Printing

- **Similar to auto-testing**
- `marking/<Assignment>/testp.<name>/`
  - `computeMarks` puts together set of files with marking scheme
- `printOut <assign> <suite>`  
`<timestamp> <piles> <where>`
  - `piles` = 'section' or number of piles,
  - `where` = -7 for `lp7_cs` , -0 for don't actually print, -g for Graphics Services Xerox 470

In order to print student electronic submissions, you will create a `testp` suite in the `marking` directory. The `computeMarks` script will use the `printFile` command to insert a copy of the marking scheme and any desired files, possibly including test results, into the marksheet that will be printed. A postscript file is then generated for each student, containing the course number, assignment number, student id, hand-box number, and all specified files. (`printFile` is documented on the ISG web site.) Note that files will have their lines truncated unless you specify that the lines are to be folded. Similarly, lines will be numbered unless you turn off the numbering.

In order to actually print the files, you should use the `printOut` command. You will usually want to print assignments on the high-speed Xerox Docutech printer (using the `-g` flag); however, since it will not accept very small print jobs, you may need to send those to `lp7_cs` in DC3116. (Note: the `piles` option is useful if you want to pre-divide the piles, for example, because you have 5 TAs marking, and each will receive a pile.)

Graphics Services guarantees delivery of the print job by 2pm the following day (same day if you print first thing in the morning), though you can pick it up personally earlier from the General Services Complex building.

## Course Email

- **When in doubt, check with instructors/ISC first (or forward to them)**
- **Carbon-copy replies so:**
  - **others know a reply has been sent**
  - **record is kept in case of problems**

Since we do not recommend that you give your personal email address to your students, students will generally email their questions to the course account instead. It is important that you read this email several times a day.

If you cannot reply right away, we strongly recommend that you tell the student so, and give them an idea of when they can expect a response.

Should the question be something you cannot answer, or is best answered by a course instructor or the ISC, please forward the email or check with them first. In fact, always check first if in doubt. You should not be making judgement calls without first checking with your ISC or course instructor.

It is a good idea to carbon-copy your replies so that others know a reply has been sent (for example, there may be more than one tutor in the course responsible for reading and responding to course email), and so that a record is kept in case of disputes/problems.

Please do not put a `.forward` file on the course account.

## Giving Other People Access

- Check first with ISC
- Automatically creates directory under `/u/csNNN/u`
  - Put person in `.rhosts` file in `/u/csNNN`
  - Rest takes care of itself when they `rlogin` if they have an account on `student.cs`; otherwise, add to `.login-passwd` file as:  
`jsmith:::Joe Smith::/xbin/tcsh`

You should rarely, if ever, receive a request for access to the course account. Since access is restricted, it is important that you first check with your ISC.

If TAs want access to certain files, it may be more secure to put them into a group such as `csNNN_ta`, and then change the file to be in the group and enable group read permissions. (Group membership can be set/modified by emailing `accounts@cs`.)

The only other people who might request access are instructors, who should have already been given access; however, there may have been problems with the granting of their access (the usual problem is that the wrong machine name was specified in the `.rhosts` file).

So, if you have to give someone access, you need to edit `/u/csNNN/.rhosts` and add the requestor's specified machine name and userid. If the requestor has an account on `student.cs`, a subdirectory under `/u/csNNN/u` will be created for them, and links to their `.cshrc` file, `.Xauthority` file, `.aliases` file, and `bin` directory will be added when they first login.

If the requestor doesn't have an account on `student.cs`, modify the `/u/csNNN/.login-passwd` file as shown above.

## Online Marks Database

<https://www.student.cs.uwaterloo.ca/~cs-marks/db/>

- **Marks are uploaded via reading in a tab-delimited text file (8-character userid column has marksheet header)**
- **Plan structure of database first since it is hierarchical, and hard to remove items.**

We use an online marks database primarily to provide the students with secure access to their marks, though it can also be used to randomly assign the students to their seats for examinations.

Since it is available through the web, it is extremely accessible.

A detailed manual is available is you need to work with the database.