Here, I intend to give a brief overview of your major responsibilities as a CS146 ISA and some technical guidelines for handling them.

THE COURSE ACCOUNT

The first thing you should do is to make sure you have access to the cs146 linux account. This is the platform on which you would do most of your work. To do this, sign into your personal student linux account and use ssh to go into the cs146 course account thus:

> ssh cs146@ubuntu1404.student.cs.uwaterloo.ca

If you are prompted for a password, this means you have not been authorized yet and you should speak to your ISC. This is the first thing you should get working since as I said most of your work will happen on this platform.

Now, you have been granted access to the cs146 account and whenever you use ssh to log in, you are allowed into the account without need to enter any passcode. Needless to say that you should be mindful of the environment in which you work so as not to let unauthorized individuals see the data stored on the account.

You can now make yourself familiar with what is on the course account. There should be a public_html folder from which you would take care of the course website. There should be a handbacks folder where you will hand-mark (if required) the student submissions, and through which you will get the scores sent back to them. More on this later.

TESTING ACCOUNT

Every course, say csXYZ has a course account csXYZ@ubuntu1404.student.cs.uwaterloo.ca and a testing account csXYZt@ubuntu1404.student.cs.uwaterloo.ca. Now the testing account is where you will prepare the submission system for each assignment. It also contains the Marmoset buildservers which test the students submissions.

To get into it, you first get into the course account, then you use ssh as in:
> ssh cs146t@ubuntu1404.student.cs.uwaterloo.ca

You should be in without needing a password. Have a look at the testing account and familiarize yourself with it.

COURSE WEBSITE

You are responsible for making sure the contents on the course website are up to date. The course website is ***https://www.student.cs.uwaterloo.ca/~cs146*** Here is where announcements, assignments and scores are posted. It also contains important information like offices and offices hours.

Have a look at it and you would see that there are probably some things that needs to be changed. For example is the term right? (You don't want it to be saying Winter2015 when the current year is 2078). What about the instructor name and office and office hours? What about your name and office and office hours?

You change the contents of the website through the course account. On the course account there should be a directory called public_html. This contains several shtml files which determine

the contents of the various webpage. Have a look at them, and perhaps play around with changing something from the file and refreshing the course webpage to see the change take effect.

You would come back to the course website several times within the term to update information. For example, to say when and where is the midterm or the final exam. The instructor at least would come in every week to post assignments.

ADDING STUDENTS TO MARMOSET

The cs146 course uses Marmoset to collect, test and grade student submissions for the various assignments. At the start of the term you would want to add all students to the system so they have access and can submit assignments. But before you can add students, you yourself need to be added.

So log onto Marmoset using your web browser at https://marmoset.student.cs.uwaterloo.ca Click into the directory that has your student name, then enter the course cs146, and at the top pane, you should see INSTRUCTOR VIEW. Click it.

If you don't see cs146, or you don't see INSTRUCTOR VIEW or INSTRUCTOR ACCESS, then it means you haven't yet been added as an instructor to the course and you should panic! No, I'm joking. Just contact your ISC.

Now you have been added to the Marmoset interface as an instructor, you have access to create projects and assignments, collect tested assignments and many more. But the first thing to do is to add other students to the system. To do this you would have to upload a classlist file to Marmoset.

On the course account, there is a file called .classlist You can see it by typing:

```
> ls -a
```

Now this classlist file contains the information of all students registeres in the course. You do not need to modify it yourself at anytime because it is modified automatically when a student is registered. To add the students to the course you get back to Marmoset through your browser, and after clicking instructor view, you see the links telling you to register students by updating a classlist file or manually. Of course, it is easier to register the students using the classlist file, so copy the contents of the classlist file to another file on your desktop (or simply download it) and upload the file to the link.

As mentioned before, students may join the course later on. In this case you can add them manually, or you can still upload the classlist file and everything is updated.

W3M

W3M is a very helpful way of browsing through the linux terminal. It can be a little clumsy to have to download a file from the student account and then upload it to the required website, or vice-versa download from the website and upload to the student account. But using W3M you can interact with the websites without ever living the terminal. You will find that this is quite helpful in creating assignment projects and in downloading student submissions. It can also be quite helpful in the previous process of uploading a classlist file.

So let us say, you want to upload a classlist file. Then from the home directory of the cs146

course account, you use W3M as a web browser thus:

> w3m https://marmoset.student.cs.uwaterloo.ca

Then using the arrow keys you can log in, navigate your way to the link, and upload the file.

PIAZZA

Piazza is a crucial discussion forum of the course and other courses of the faculty. Log onto it to make sure you have access to the cs146 course. If you do not, then talk to your ISC so that you are added.

You should listen to the Piazza discussions frequently to address the questions and followups made by the students.

COURSE EMAIL

The course email is cs146@uwaterloo.ca Students will contact you for course related issues through this email. So talk to your ISC to get the password for this email, and perhaps set up forwarding so that all mails sent to it show up on your personal email. But more importantly, always reply to mails through the course email and never through your personal email.

ASSIGNMENTS

One of your recurring duties would be working on the assigned weekly assignments. Of course you would have other duties such as listening to Piazza discussions, checking for emails, holding office hours, teaching tutorials, attending lectures etc.

Now assignments are posted approximately every week, and it is your duty to start work on the assignments early, prepare the Marmoset project through which the students submit their work, make up tests for the assignments, and after the deadline, if there is to be handmarking, collect the assignments and grade them for the required hand-mark qualities.

At this point, what you may want to do is to setup an excel document containing all user ids and after every assignment is due, enter the student grades into the document. This would save you a lot of work when the term comes to a close and you want to bring things together.

So about working on assignments, as I said you should start early so you can have things setup on time for the students. If you feel you may need more time, you may want to arrange with the instructor to have the questions released to you before they are made public to the students.

CREATING ASSIGNMENT TESTS

Once you have the question(s), your target is this:

1. Create a solution for the assignment
2. Create the tests upon which student submissions are checked
3. Create a project for the assignment through Marmoset
4. Upload the test setup to Marmoset
5. Activate the assignment

Let us go over the first two using an example. Let's say this is the first assignment:

Write a function num_divisors that takes a positive integer and returns its number if divisors. File to submit: Q1.rkt

Step1 is to create a solution, for example the below file:

```
#lang racket

;; (num_divisors n) takes a positive integer n and returns its number of positive divisors
;; num_divisors : Num -> Num
;; Requires : n > 0

(define (num_divisors n)
	(define (num_divisors/acc m div count)
		(cond [(< m div) count]
			[(= 0 (remainder m div)) (num_divisors/acc m (add1 div) (add1 count))]
			[else (num_divisors/acc m (add1 div) count)]))
	(num_divisors/acc n 1 0))
```

Step2 is to write tests by which student code will be accessed. You will create public tests and release tests (and possibly also secret tests). The public tests check the very basic requirements. The release tests go in depth, checking edge cases etc.

So let us say you want to create 1 public test and 2 release tests. Then for step2 you will have to create 9 files:

- public0
- public0.in
- public0.out
- release0
- release0.in
- release0.out
- release1
- release1.in
- release1.out

You see that for each test T, you create 3 files T, T.in, T.out. The file T.in contains the input that is passed into the program. The file T.out contains the output expected if the program is correct. The file T is a bash file that when executed, takes the program and tests it against T.in and T.out.

So as an example, T.in may simply be the below file:

```
6
```

And T.out would be the file

```
4
```

Then T would be a bash file. Now there are many ways to write the bash file. You just need basic bash programming skills. One idea is to take the content of the student submission and copy it

into a new temporary file, then add at the end of the temp file, the line (num_divisors ***) where *** is the input in T.in, then run the temp file and compare the result using the racket equal? Function. There are more robust ways to write the file T and you can have a look at previous terms for guides.

Finally, I should mention that for a given test T, it is not necessary to have all three files T, T.in and T.out. It all depends on how your test is structured. Only file T is necessary since that is the file Marmoset would call in testing the student submission. You may for example omit T.in and T.out and incorporate them in file T itself, or you may omit T.in alone, or T.out alone. It all depends on you. Again have a look at previous tests if this is not clear.

CREATING THE MARMOSET PROJECT

The final step in setting up the assignment submission system is to create a project. You do this through the Marmoset web interface. Once you are on the instructor view, you would see a "Create New Project" link which you can click and it takes you to a table where you fill in the information for the project such as project name, deadline etc.

For example, for the project number you may have Q1, for the ontime deadline, you may have 10 January 2078, 10:00pm. You would set the late deadline to the same thing if you do not want to accept late submissions. For the project title you may have "Number Of Divisors". I always left the url field empty but you can put anything here like a link to the assignment etc. For the description field I usually put information of what file the students are to submit. Hence in this case I would put: "File to submit: Q1.rkt"

For the stack trace policy, I set it as the last one – full output printed to stdout. The rest fields I usually leave as default although you ca change them. For example how many release tokens do you want the students to have? How long before tokens regenerate? Etc.

Now the project is created, the final step is to upload the test setup then activate the project. There are two types of setup: static test setup and dynamic test setup. You can read about them in depth here https://cs.uwaterloo.ca/twiki/view/ISG/Marmoset I usually use the dynamic test setup because it is easier to create and easier to manage. The basic idea is that for the static test setup, you take all the test files and script and you upload it to marmoset. But for the dynamic test setup, you don't do that. You just upload a single file that tells marmoset where the test files are located. The dynamic test setup is also easier to manage. If something goes wrong with the static test setup, you have to upload a new set of test files and scripts, but with the dynamic test setup you just change whatever is wrong from the student account and you dont need to upload anything.

So to create the dynamic test setup the file you upload is test.properties. Well, not quite. You create the file test.properties and then you zip it into for example in this case you zip it into Q1.test.zip. You know how that is done right?

> zip Q1.test.zip test.properties

The test.properties contains information about what tests are used and where the tests are located. For our example, this would be what the test.properties file would look like:

```
build.language=UW
test.class.public=public0
test.class.release=release0 release1
build.make.command=/u6/cs146t/marmoset/bin/dynamic_test
```

build.make.file=/u6/cs146t/marmoset/ASSIGNMENTS_FOLDER/Q1

Having created this file and zipped it, you go to the created project and click the utilities tab and you will see where is said “Upload new test setup” and this is where you throw in the zip file. But remember you can do all this through the terminal using W3M.

Now you have thrown in the test setup, one last thing is to throw in the canonical solution. This is your own solution to the problem created at the beginning. You take your solution Q1.rkt and zip it into say Q1.sol.zip and you upload it via the link in the utilities page which says “Project submission ... canonical ...”. Marmoset will test your solution against the test setup and if it passes it, it allows you to allocate points for the tests. So for example we may give public0 1 point, release0 3 points and release1 6 points. It all depends on you. After allocating the points then you go to the bottom of the utilities page to make the project visible.

SOME COMMENTS

Creating the assignment projects is not hard and you get the hang of it after doing it once or twice. In any case refer to the ISG twiki or speak with other ISAs or your ISC if anything is unclear. One comment I should make is to make sure your solution is not on the assignment folder, otherwise all submissions would pass the test because instead of the submission being put on the folder, it sees that yours is already there and it uses yours.

Another thing is that if students complain that Marmoset is not working for them, then you may want to look at the link in Marmoset that says BUILDSERVERSTATUS to see that no build servers are hanging. If any are hanging and have been hanging for a while, then you may want to restart them. The instructions for doing that are here:
https://cs.uwaterloo.ca/twiki/pub/ISG/Marmoset/RestartingMarmosetbuildservers1.pdf

HANDBACKS

The cs146 course uses the handback system to return scores to the students. You can see the handback link on the course website. The basic idea is that when you click the link and you enter your quest credentials, if you are a student of the course or an instructor or ISA, you are allowed in and you can see the list of directories, one directory per student. But only the student who owns the directory (or the instructor or ISA) can view the contents.

How does this work? Well, on the course account, in the public_html folder, you would see a handbacks folder. In this handbacks folder you should have (if not you create it) a .htaccess file. This is the file that says who is authorized to see the contents of the handbacks folder. The content of the .htaccess file is as such:

```
SSLRequireSSL
require user prof
require user isa
require user daffy
require user bugs
require user lola
```

prof, isa, daffy, bugs, lola etc are the waterloo usernames of the individuals who can view the contents of the handbacks folder. Any other person would not be allowed. The same idea works for the folders inside the handbacks folder. For example there would be a folder named daffy for the

student Daffy Duck and it would have it's own .htaccess file which may look like this:

```
SSLRequireSSL
require user prof
require user isa
require user daffy
```

This means that only the instructor, the ISA, and Daffy Duck can view the contents of this folder.

Now the handback system is an important way of giving individual feedback to students. For example some assignments may be marked for style and then you send the results and comments back to the student using the handbacks folder. There is a folder on the course account proper called handback. You can do the marking here. That is, you download the assignments from marmoset (Remember W3M). You get all assignments into a folder say Q1. You do the marking in Q1. Now in the handback folder, you will see two curious bash files handback.sh and cutEnd.sh. Now, when you are done marking or commenting, running handback.sh on the folder which contains the files you want to give back to the students, would send the files to the handbacks folder (in public_html) so the students can see them.

> ./handback.sh Q1

But for this to work as you want each file to be sent should be in a folder with name equal to the name of the student. Hence for example, the handback folder contains the folder Q1, then the folder Q1 contains folders daffy, bugs, lola. Then these three folders contain the file Q1.rkt. So now when you run “./handback.sh Q1”, the files Q1.rkt are added to the viewable collection of the students. The second curious file is cutEnd.sh but really what it does is simple. When you download the submissions from marmoset, the name of the folders would have additions. So instead of daffy, the folder would be named daffy-On-time. Instead of lola, it would be named lola-On-time and so on. CutEnd.sh just fixes this silly issue.

> ./cutEnd.sh Q1

ARCHIVING

Needless to say that being organized would go a long way in making your work term go smoothly. At the end of the work term, you should clear up what is not needed for the next term, and save appropriate materials. Look at the previous terms or contact your ISC to find out what should be archived and what should be deleted.