COMP2100 Lab Command Manual

Len Hamey Revised 22 July 2020 len.hamey@mq.edu.au

Note: Output of the lab command shown in this document is for illustration purposes only. The details of individual assignments may differ from the examples shown here.

Introduction

The unit COMP2100 incorporates labs that are automatically generated and automatically marked. The lab command is used to operate these labs (unless otherwise specified). The lab command provides for fetching assignment material, submitting solutions, accessing marks and managing extensions. The lab command can be found at

/home/unit/group/comp2100/lab

The lab command must be run on the student servers ash or iceberg – you cannot copy the lab command to any other Linux machine, nor will it work correctly on any other machine because the server that it connects to only permits connections from the specified servers.

Simplifying the lab command

It is tedious to type the full path to the lab command every time. This section explains how to simplify the command so that you can type 'lab' instead of '/home/unit/group/comp2100/lab'.

Commands in Unix are interpreted by a "shell" command interpreter. We will learn more about how the shell works towards the end of the unit. For now, you just need to learn how to use the Unix command line, i.e. the shell.

The particular shell that we use on ash and iceberg is called "bash". When you log in, the system starts up the bash shell for you to type commands into. Whenever bash starts up, it reads a file in your home directory called ".profile" and executes the commands that it finds there. The dot (.) at the start of the file name hides the file when you list the directory, but it is still there. Using .profile, you can tell the shell to perform certain initialisation every time it starts up. We will use this facility to simplify the lab command.

The shell has a command "alias" which sets up a short name for a longer command. Whenever you type the short name, the shell recognises the alias and executes the longer command instead. We will use an alias command inside the .profile file to simplify the lab command.

Here is the alias command that we need:

```
alias lab=/home/unit/group/comp2100/lab
```

We need to put this command into .profile. The best way to do this is to edit the file .profile and type the command into the file on a line by itself – do this if you know how to use a unix editor. However, if this is your first experience with unix, you may not yet know how to use an editor, so here is a simple procedure that you can follow.

1. The following command appends the alias command to your existing .profile file, or creates a new one if you do not already have one.

Carefully type this command **exactly** as shown below. If you make a mistake, you should seek help to repair your .profile file. Note that the \$ sign represents the Unix system prompt. The command itself is the underlined text – type only this text and press Return to execute the command.

```
$ echo "alias lab=/home/unit/group/comp2100/lab" >> .profile
```

- 2. Log out and log in again.
- 3. Try the "lab" command. Here is what should happen. If it does not, then you will need to go back to step 1, ask for help and correct the mistake.

```
$ lab
Usage: lab [-g|-s|-sf|-m|-d|-x] lab.stage [file ...]
-g lab.stage: Get lab stage.
```

Running the lab command

To execute the lab command, type a command similar to the following Unix command. Note that the \$ sign is the Unix prompt – you should only type the command that is underlined in this example. In future, we won't bother to underline the command since you are now familiar with the \$ prompt notation.

```
$ lab
```

When you run the lab command without any parameters, it prints out brief usage information, as is common for many Unix commands. This information is provided to remind you of how to use the command – it lists the available options and their meaning.

Workflow

The detailed workflow for COMP2100 labs depends on the particular lab and will be explained in the lab description. Typically, you will fetch one or more files using the lab command. These files will define your particular tasks for the lab, which differs in various ways from the task set for other students. Using those files, you will work on the lab, developing your own solution. Depending on the lab, you may use the lab command to submit your solution for automarking. You can resubmit as often as you wish until the lab closes. After the lab is closed, it may still be possible to continue to work on it and submit solutions, but any work you do will not count towards your mark unless you are granted an extension for reasons of special consideration.

Lab Achievement Marks and Progress Marks

Each lab assignment includes two separate sources of marks: The marks for your achievements in the lab, and progress marks for working consistently throughout the lab.

Achievement marks. The main marks are your achievement marks. An achievement mark reflects the quality of your work in each stage of the assignment. Individual assignments may have other achievement marks such as the programming style marks in the Data File Lab. See the assignment specification for details.

Each lab has a closing date. Until that date, you are free to work on any of the lab stages to improve your achievement marks in the lab.

Progress marks are awarded for showing evidence of progress in the lab each week (or earlier). For example, in Data File Lab there is a progress mark that is awarded for showing progress at the beginning of week 4, and there are other progress marks for other weeks. To be objective, we need a clear definition of "progress" so we list some milestones that you can reach and we define "progress" as reaching a new milestone each week (or earlier). Each milestone typically corresponds to a stage of the assignment, and you get the process mark if your achievement mark for that stage reaches the required level such as 2 marks out of 3. See the assignment specification for the exact specification of the progress requirements.

In order to be awarded the progress mark for a particular week, you need to reach a new milestone no later than the milestone due date and time in that week. If you miss the milestone for a particular week, then you don't get the progress mark for that week but you can earn the progress mark of a later week when you finally reach the original milestone.

For example, consider the following two students: Susan and Robert. We show what stage each student is working on for the milestone in each week, and whether they reached the milestone require or not.

	Susan		Robert		
Week	Milestone	Progress	Milestone	Progress mark	
	reached?	mark	reached?		
4	Stage 1: reached	1	Stage 1: not	0	
			reached		
5	Stage 2: reached	1	Stage 1: reached0	1	
6	Stage 3: reached	1	Stage 2: reached	1	
7	Stage 4: reached	1	Stage 3: reached	1	
Total progress mark		4		3	

Susan has reached a new milestone each week, so she earns 1 progress mark each week and ends up with the maximum possible progress mark total. Robert does not reach the stage 1 milestone in week 4, so he misses that progress mark. However, when the week 5 progress date arrives, Robert has reached the stage 1 milestone so he is awarded the week 5 progress mark for his work on stage 1. By this time, he is probably working hard on stage 2, so when the week 6 progress date arrives, Robert has reached the milestone for stage 2 and he is awarded another progress mark. Similarly, he is awarded a progress mark in week 7 for his work on stage 3. In the end, Robert has slipped behind by one week and has lost one progress mark. Missing a single progress deadline costs you only one progress mark. However, falling behind is likely to cost you many more achievement marks because the lab is not designed to be completed in a rush – you are unlikely to complete the lab if you do not reach or exceed the progress milestones each week. In the example above, Susan achieves good outcomes in four stages but Robert has only achieved good outcomes in three stages.

The lab command clearly indicates when a progress mark is awarded. Once the progress mark has been awarded, you can continue to work on the lab stage without risk of losing the progress mark, even if your achievement mark goes down due to new bugs.

For those students eager to complete the lab early, progress marks can be earned ahead of time. For example, if you complete stages 1, 2, 3 and 4 by the third progress date then you will be awarded all four progress marks because you reached the milestones ahead of time.

Usage

Run the lab command without any command line parameters to obtain brief usage information. Here is what the information looks like.

```
$ lab
Usage: lab [-g|-s|-sf|-m|-d|-x] lab.stage [file ...]
  -g lab.stage: Get lab stage.
   -s lab.stage:
                             Submit lab stage solution.
  -s lab.stage:

-sf lab.stage:

Force submit even if the mark is reduced.

Report summary marks for all labs.
   -m lab:
                             Report marks for a specific lab.
  -m lab.stage: Report marks for specific lab stage.
   -d:
                             Report due dates including extensions.
                             Report your extensions.
   -x:
   -x lab.stage days: Claim or change extension.
   lab.stage: numeric lab number and stage number.
Examples:
   lab -g 1.2
                    Get stage 2 of lab 1
   lab -s 1.2 main.c sub.c include.h
                     Submit solution for stage 2 of lab 1
  lab -m
Show marks for labs managed by lab command
lab -m 1
Show marks for lab 1 by stage and progress
lab -m 1.2
Mark report for stage 2 of lab 1
lab -d
Report all due dates
lab -d 1
Report due dates for lab 1
lab -x
Report of all extensions granted
lab -x 1 2
Claim free 2 day extension of lab 1 closing
   lab -x 1.3 2 Claim free 2 day extension of lab 1 progress 3
```

There are 6 command-line options listed. You always have to select one of the command-line options to indicate what you want the lab command to do for you: get a lab stage, submit your solution to a lab stage, forced submission, display a marking report, display due dates or manage extensions.

Get a lab stage

Use the -g option to download the files for a lab stage. The lab command stores the downloaded files in the current working directory, and reports their name(s). If there are no files to be downloaded, the lab command will tell you. Typically, the server may actually send you a tar file that should unpack to obtain all the files that you require. See man tar for information on tar files.

There is only one parameter associated with the –g option – it is the lab stage identifier in the format *lab.stage*. For example, the following commands fetch and unpack lab 1 stage 1. The unpacked files are placed in a subdirectory called stage1.

```
$ lab -g 1.1
Downloaded stage1.tar
$ tar xvf stage1.tar
stage1/
stage1/initialisation-specification.txt
stage1/marking-guide.txt
stage1/expected-output.txt
stage1/filestruct-description.txt
```

Submit your solution to a lab stage

The -s option submits your solution to a lab stage. Use this option to submit your solution for immediate automatic marking. You must list all the files that you wish to submit – each submission is completely separate from any previous submissions. For example:

```
$ lab -s 1.1 stage1.c
Uploaded 1 file
--> ./main
look, opinion, downtown, quartz, suggestion, leg, brick, dirt,
face, ring, sweater, badge, oatmeal, airplane, leaf, lumber,
stone
194, 1, ffffffffffffffa8d, H, 0.042236, 1, 0, 0, 1714.385254,
30, -7, 85, 75, -28, fireman, -93.489006, 1
MARKING REPORT:
                                          1.0 / 1.0
Correct output
                                          0.9 / 0.9
Correct struct definition
                                          0.8 / 0.9
Correct initialisation of struct
Exit without error in normal usage
                                         0.2 / 0.2
TOTAL
                                          2.9 / 3.0
ERRORS:
Check initialisation specification: sweater
**** Previous mark for this stage was 3.0
**** Submission not saved. To force update use lab -sf
Progress for stage 1.1 was previously achieved.
Your submission for stage 1.1 (from a previous session) will
be marked for style.
```

The command above submits stage1.c as the solution for lab 1 stage 1. The server will receive the file and then automark it. In this example, the file is compiled and the program (called ./main) is executed and tested to see how it performs. The display shows the output of the program (two lines of text that are wrapped on the screen) followed by a detailed marking report. The report provides a breakdown of the marks and identifies where the error has been found.

For each lab stage, the server keeps a record of the *final mark* achieved and also the *final submission*. Whenever you submit a new solution, the server compares the new mark with the final mark for that lab stage. If the new mark is at least as high as the final mark then the server replaces the *final submission* with the new submission and updates the final mark for the stage. However, if the new mark is less than the final mark, the server does not update either the final mark or the final submission. In this way, the final submission corresponds to the submission that achieved the best final mark.

The lab command clearly tells you if your submission is not being saved as the final submission. In the example above, the two lines highlighted with asterisks remind you that the current final mark is 3.0 (from a previous submission) and that the new submission is not being saved.

Two final information messages indicate whether you have been awarded a progress mark for the achievement in this submission, and which submission is currently the one that would be marked for

program style. Program style is a feature of this particular lab – other labs may have different informational messages.

Forced submission

In rare situations, you may consider that your new solution is more suitable as the final submission even though its achievement mark is not the best. For example, if you have been improving program style then you may want the server to accept your improved program even if a small bug has been introduced. The -sf option provides this capability. For example:

```
$ lab -sf 1.1 stage1.c
Uploaded 1 file
--> ./main
look, opinion, downtown, quartz, suggestion, leg, brick, dirt,
face, ring, sweater, badge, oatmeal, airplane, leaf, lumber,
stone
194, 1, fffffffffffffffa8d, H, 0.042236, 1, 0, 0, 1714.385254,
30, -7, 85, 75, -28, fireman, -93.489006, 1
MARKING REPORT:
Correct output
                                          1.0 / 1.0
Correct struct definition
                                         0.9 / 0.9
                                          0.8 / 0.9
Correct initialisation of struct
Exit without error in normal usage
                                         0.2 / 0.2
                                          2.9 / 3.0
TOTAL
ERRORS:
Check initialisation specification: sweater
---- Previous mark for this stage was 3.0
--- New submission has been accepted
Progress for stage 1.1 was previously achieved.
Your submission for stage 1.1 (the program you just submitted)
will be marked for style.
```

This command operates almost identically to the command shown earlier that uses the submit option. The only difference is that, when the server has marked the submission and found that the mark is less than the currently recorded final mark, it prints a warning information message highlighted with dashes, accepts the submission as the final submission and <u>updates the final mark to the lower mark</u>. In this case, because stage 1.1 will be marked for program style, the new final submission would be marked for style, provided that there are no other future submissions that change the server's selection.

Warning: Forced submission is risky since you will <u>lose marks</u> for the stage and your final submission may be seriously flawed. You should only use forced submission as a last resort, when the lab is about to close and you desperately want to submit the latest version of your solution because you believe that it will earn you more marks (perhaps for style) than the older more correct solution.

Backup and version control: Good practice is to maintain backup copies of all your work, and to use version control to manage your changes. Backup protects you from disasters such as losing your USB drive or a hard disk failure. Version control protects you when you make mistakes such as introducing a bug that you cannot easily find – you can compare your current version with a previous working version to see where the bug may have been introduced; and, in the worst case, you can roll back to the previous working version and pick up from there again.

Backup and version control are highly recommended. For backup, keep recent copies of your work on the disk server claudius which is backed up by IT staff and will be restored if there is a hard disk failure. Your home directory in ash/iceberg is on Claudius, and so is your H drive in the Windows machines in the labs. Files kept in these places are protected against hardware failures.

For version control, see Git or similar tools. You will use Git in other units, so it's worthwhile investing in learning it. A poor (but minimal) approach to version control is to periodically make a local copy of your project as a tar file (Unix) or zip file (Windows). These "snapshots" protect you from major damage to your code but don't help with debugging. If you are enrolled in COMP2020 Software Engineering you will be learning about Git in week 2 and we encourage you to use Git for COMP2100 also. Please be sure that your Git repository is kept private.

If using both backup and version control, be sure to back up your entire repository, not just the most recent source code checkout version. The best way is to keep your repository on Claudius.

Backup and version control are your responsibility. The lab command does not function as a backup facility and you cannot fetch previous submissions.

View a report of your marks

Whenever you successfully submit a solution to a lab stage, the lab command displays a detailed marking report, as in the examples above. However, you can view your marks that are managed by the lab command at any time using the -m option.

There are three ways to view your marks: An overview of all labs, a summary of a single lab or a detailed report of a single lab stage.

All marks overview

To obtain an overview of your marks, use the -m option without any other parameters as in the following example.

\$ <u>lab -m</u>			
MARKING :	SUN	MARY	REPORT
Lab		Mark	(S
Lab 1		8.9)
Progress	1	3.0)
Lab 2		2.5	5
Progress	2	1.0)
Total		15.4	1

This example provides an overview of marks for a student who has completed some of lab 1 and has made a start on lab 2. For each lab, the command reports the lab stage marks achieved and the progress marks. Note that Lab 1 style marks are not included in this example report.

Single lab summary

To view a summary of an individual lab, specify the lab number as the parameter after the -m option. For example:

\$ <u>lab -m 1</u>							
MARKING FOR	LAB 1						
Stage	Marks						
1.1	2.9	2016-07-11	14:03:54				
1.2	3.0	2016-07-15	12:40:07				
1.3	3.0	2016-07-18	15:11:36				
1.4	3.0	2016-07-22	18:12:54	(skipped)			
1.5	4.0	2016-07-29	12:23:02				
Progress	Marks						
Aug-09	1.0	2016-07-09	18:41:16	awarded	for	stage	1.1
Aug-16	1.0	2016-07-13	10:38:44	awarded	for	stage	1.2
Aug-23	1.0	2016-07-16	10:40:29	awarded	for	stage	1.3
Total	15.9						

Here we see a detailed report for lab 1. The report shows the marks achieved in each stage of the lab, and the date and time when each lab stage was last submitted. The progress marks are also listed in detail, showing the date and time when each progress mark was earned and the stage that was awarded the progress marks. In the report, progress marks are identified by the date on which they fall due.

Dates and times are presented in Excel format: year-month-day hour:min:sec where time is shown on the 24-hour clock. This format is used because it is easy to import reports into Excel for analysis.

Notice that the lab report indicates that stage 1.4 has been "skipped". This is a feature of lab 1, where the final total mark is computed by choosing the greater of the marks for stages 1.4 and 1.5. The mark that is not included in the total is "skipped" – it is stage 1.4 in this case. Other labs may have their own special ways of computing the total lab mark, as explained in the specific lab assignment documentation.

Single lab stage detailed report

You can view the detailed marking report for a single lab stage by specifying the lab stage as the parameter of the -m option. A lab stage is specified as the lab number, followed by period, followed by the stage number. For example:

```
$ lab -m 1.1
DETAILED MARKING FOR LAB 1 STAGE 1
MARKING REPORT:
                                         1.0 / 1.0
Correct output
                                         0.9 / 0.9
Correct struct definition
                                         0.8 / 0.9
Correct initialisation of struct
Exit without error in normal usage
                                        0.2 / 0.2
                                          2.9 / 3.0
TOTAL
ERRORS:
Check initialisation specification: sweater
Note: To view progress marks, request lab report or all labs
report
```

The single stage report consists mainly of the detailed marking report that was displayed when the submission was originally marked. Indeed, the detailed report from that time was saved by the server and can now be presented again to you. The information message refers you to other reports to view the progress marks that may have been awarded for achievement in this lab stage.

Report of due dates

The -d option reports due dates. Without any other parameters, it reports all due dates known to the lab command. The dates are reported in a user-friendly format. If there are any active extensions, they are shown and the due dates reported are the extended due dates. In the following example, the user has claimed a free extension day on the closing date of the first lab. (Note: The due dates vary from year to year, so the details of this example are not relevant to the current year).

You can obtain a report for a specific lab by specifying the lab number as the command-line parameter after the -d option. For example:

Extensions of due dates

The -x option manages extensions of due dates. You can use this option to view your active extensions, and to claim free extensions. The lab command implements a limit of 3 free extension days throughout the semester. You can claim free extensions for the closing dates of labs and/or for progress dates. However, extensions of progress dates are unlikely to benefit you as much as extensions of the closing date. This is because extending a progress milestone does not extend the following milestones, so you should work hard to achieve the progress milestones on time and not rely on extensions. Your 3 free extension days are a valuable resource which you should manage to your best advantage. For more information about free extensions, see the Unit Guide.

The lab command also implements extensions that arise from special consideration — when illness or misadventure affects your ability to meet the progress milestones or the lab closing date. If you believe that you have a case for special consideration, follow the procedures described in the Unit Guide. Be sure to notify the unit convenor as quickly as possible. We can grant extensions through the lab command that will have immediate effect. You can view your special consideration extensions (which the lab command calls "authorised extensions") and your free extensions using the -d option or the -x option.

View your active extensions

With no other parameters, the -x option reports your active extensions. Unlike the -d option which reports all due dates whether they are extended or not, the -x option reports only your active extensions.

```
$ \frac{\text{lab} - x}{\text{Lab} + 1} Lab 1 closing time extended by 1 free day to Tue 11 Sep 2018, 12:00:00. You have 2 free extension days remaining.
```

This example report shows that a free extension has been taken. You can manage your own free extensions – in this example, the student has previously claimed a free extension of 1 day on the closing of lab 1. The report shows that the student has 2 more free extension days yet to be claimed.

Although it is not shown in this example, you can combine a free extension with an authorised extension, and the two extensions will be added together. So, if you are granted special consideration due to misadventure near the closing of a lab and you are granted an extension of 2 days, you can still use a free extension to gain an additional day if you have free extension days available.

Claim or change a free extension

To claim or change a free extension, specify the lab stage that you want to extend and the number of days extension that you want to apply. To claim an extension of the lab closing date, specify only the lab number but to claim an extension of a progress milestone, specify the *lab.stage*.

If you specify a lab closing or progress milestone that already has a free extension claimed, the effect of the lab command is to change the number of days to whatever you specify on the command. In particular, if you wish to cancel a free extension, specify 0 as the number of days. You can make any changes you like to a free extension, except that you cannot exceed 3 days claimed in total, and you can only make or cancel future claims. (To be precise, it is not possible to make or change a free extension claim so that the new extended due date is in the past.)

Here is an example of claiming a free extension. This example follows from the example above. The command changes the extension of the closing date of lab 1 from 1 day to 2 days. The lab command displays an updated report of all extensions, highlighting the particular item that has been changed. It also reminds you how to cancel an extension – note that "1.close" is another way of indicating that you want to change the closing date of lab 1.

```
$ lab -x 1 2
MODIFIED: Lab 1 closing time extended by 2 free days to Wed 12
Sep 2018, 12:00:00.
You have 1 free extension days remaining.

To cancel your new free extension, use the command:
    $ lab -x 1.close 0
```

In the following example, the user cancels the extension of lab 2 closing time. In this example, the cancellation was permitted because the date and time was no later than 10 Sep 2018, 12 noon which was the original due date for lab 1 in 2018. Had the date and time been later, the cancellation would not have been permitted.

```
$ \frac{\text{lab} - \text{x} \cdot 1 \cdot 0}{\text{MODIFIED:}} Lab 1 closing time extended by 0 days to Mon 10 Sep 2018, 12:00:00.
You have 3 free extension days remaining.
```

A final warning: Be careful when you claim free extension days because it is possible to make a claim which you cannot cancel. Suppose that the lab has closed 1 hour ago and you claim a 1 day free extension for the lab that has just closed. Provided that you have sufficient free extension days remaining, the extension will be granted and you will be able to submit the lab until the extended due time. However, it would not be permitted to cancel the extension because the original (unextended) due time has passed. The take-home message here is: use your free extensions wisely and carefully.