

PROGRAMMING ASSIGNMENT– 02

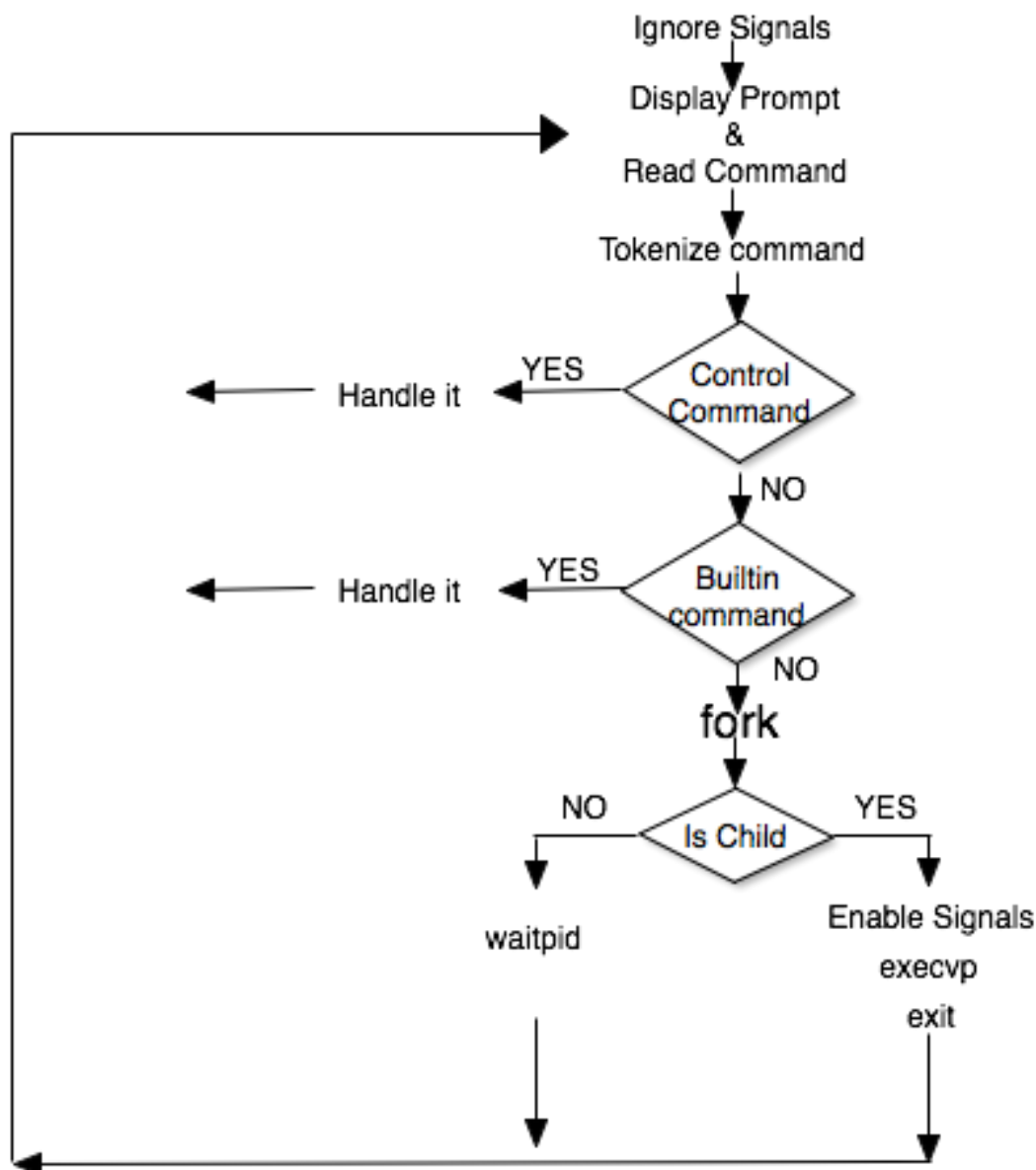
Advance Operating System

Creating UNIX Shell (100 marks)

Problem Statement

The purpose of this assignment is to give you practice in the use of UNIX system calls and writing a command interpreter on top of UNIX. The Shell as described by Richard Stevens in his book Advanced Programming in the UNIX environment is a command-line interpreter that reads user input and execute commands. For some it is a user interface between user and the internals of operating system whose job is to intercept user's command and then trigger system calls to ask OS to accomplish the user's tasks. For me it is a program executing another program.

A shell mainly consists of two parts: parsing user requests and accomplishing user request with system call's help. In this assignment you will write your own command shell to gain experience with some advanced programming techniques like process creation and control, file descriptors, signals, I/O redirection and pipes. You will do increment programming and develop different version of your own UNIX shell whose specifications are mentioned in the following paragraphs. A simple flow chart of the expected shell is given below:



Version01:

The first version of **shell** has been discussed in the class and developed in the video lecture available online at (<http://www.arifbutt.me/lec22-design-code-unix-shell-utility-arif-butt-pucit/>). It has the following capabilities/characteristics:

- The shell program displays a prompt, which is **name@pwd**, i.e., the present working directory. You may like to indicate other things like machine name or username or any other information you like. Hint: **getcwd()**
- The **shell** allow the user to type a string (command with options and arguments) (if any) in a single line. Parse the line in tokens, **fork** and then pass those tokens to some **exec** family of function(s) for execution. The parent process waits for the child process to terminate. Once the child process terminates the parent process, i.e., our **shell** program again displays the prompt and waits for the user to enter next command.
- The shell program allows the user to quit the shell program by pressing **<CTRL+D>**
- If the user run the **less /etc/passwd** program and press **<CTRL+C>**, i.e., send **SIGINT** to the **less** program. Both the **less** program and the shell should not terminate. Rather the signal should be delivered to the **less** program only and not to **shell**

```
$ ./shellv1
shell@/home/arif/:- ls -l /etc/passwd
-rw-r--r-- 1 root root 2102 Feb 7 07:35 /etc/passwd
shell@/home/arif/:-
```

Version02:

This version should be able to redirect **stdin** and **stdout** for the new processes by using **<** and **>**. For example, if the user give the command **\$mycmd < infile > outfile**, the shell should create a new process to run **mycmd** and assign **stdin** for the new process to **infile** and **stdout** for the new process to **outfile**. Hint: Open the **infile** in read only mode and the **outfile** in write only mode and then use **dup2()**. For your ease, you may assume that each item in the command string is separated on either side by at least one space. This version should also handle the use of pipes as we all are used to of it. For example the first sample command will create a copy of **file1.txt** with the name of **file2.txt**. The second command will print the number of lines in the file **/etc/passwd** on **stdout**.

```
$ ./ shellv2
shell@/home/arif/:- cat < file1.txt > file2.txt
shell@/home/arif/:- cat /etc/passwd | wc
96      265      5925
shell@/home/arif/:-
```

Version03:

This version should be able to place commands (external only) in the background with an **&** at the end of the command line. Running a process in the background means you start it, the prompt returns at once, and the process continues to run while you use the shell to run other commands. This one sounds tricky, but the principle behind it is surprisingly simple. Need to handle signals and avoid zombies. Sounds like an adventure movie. The regular shell also allows the user to separate commands on a single line with semicolons. This version should also support executing multiple commands on a single line if a semicolon separates them (;). Hint: try **waitpid(pid, status, options)**

```
$ ./ shellv3
shell@/home/arif/:- find / -name f1.txt &
[1] 1345
shell@/home/arif/:- echo "PUCIT" ; date ; pwd
PUCIT
Sat Apr 16 18:27:24 PKT 2020
/home/arif
shell@/home/arif/:-
```

Version4 (Bonus):

Built in commands are different from external commands. An external command is a binary executable, which the shell searches on the secondary storage and then do **fork** and **exec** to execute it. On the contrary the code of a built in command is part of the shell itself. So before calling **fork** and **exec**, you have to see if the command is built into the shell. This version should allow your shell program to use some of the built in commands like

- cd**: should change the working directory
- exit**: should terminate your shell
- jobs**: provide a numbered list of processes currently executing in the background
- kill**, should terminate the process numbered in the list of background processes returned by jobs by sending it a **SIGKILL** signal. Hint: **kill(pid, SIGKILL)**
- help**: lists the available built-in commands and their syntax

Submission Instructions:

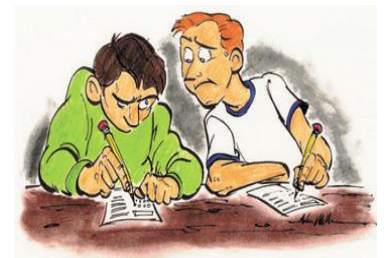
- **What to Submit:**

- Source Code files:
 - Design your code to be in multiple .c files for better understanding.
 - Your code needs to be properly commented and have necessary error checking.
- README file:
 - There should be a README file that should state your code status and bugs you have found
 - List the features you have implemented.
 - Acknowledge the helpers or the Internet resources if you have.

- **How to Submit:**

- Create a private repository named **assignment02_rollNo** (all lower case) on you own bitbucket account
- Make your course instructor (arif@pucit.edu.pk) and course TAs the member of your repository (Access level minimum read).
- You have to commit your code on created repository after completion of every version of above programs. Every version must have its own README file describing all the functionalities of the code.
- **Deadline for uploading the assignment on bitbucket account is 11:59 pm Saturday, January 30, 2021.**
- Any cheating case will result in a zero mark in this assignment and may be a grade down in the overall course as well.

TIME IS JUST LIKE MONEY.
THE LESS WE HAVE IT;
THE MORE WISELY WE SPEND IT.
Manage your time and Good Luck

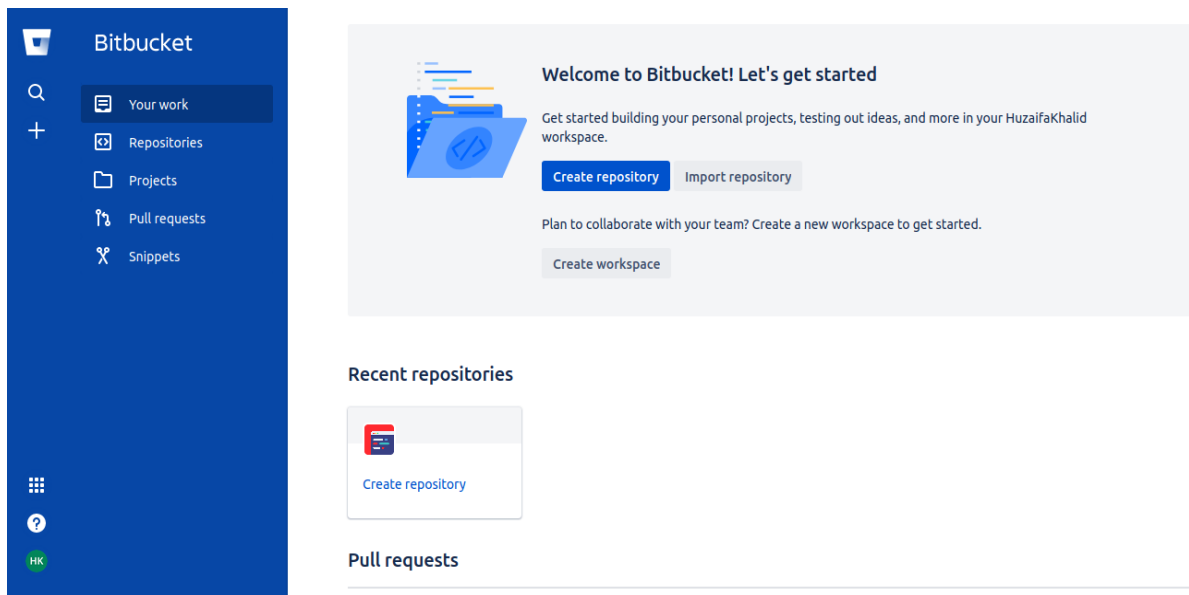


Submission Instructions (For Assignments)

Creating Bitbucket Account

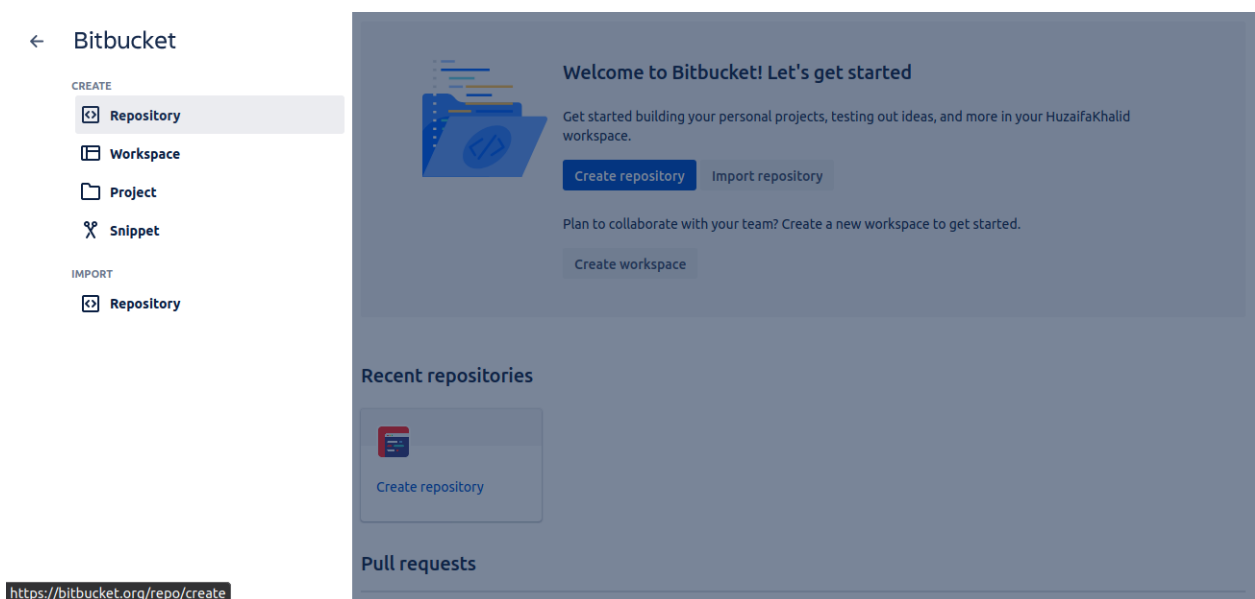
- For Assignment submission, we will use bitbucket which is a web-based version control repository hosting platform. Go to bitbucket website (<https://bitbucket.org/>) and create an account.

- After signing in, this dashboard will appear in front of you.



Creating Repository

- Now you have to create a repository. For this, click + in the global sidebar at left and then under



create, select **Repository**.

- Following screen will appear in front of you. Write a project name and a repository name. Repository name should be **AOS_Assignment#_yourRollnumber**. Access level should be private. Leave everything else as it is. After that click **Create repository**.

Create a new repository

[Import repository](#)

Workspace

ZARAFSHAN AJMAL

Project name*

AOS_ASSIGNMENTS

Repository name*

AOS_Assignment01_mscsf19m015

Access level

☒ Private repository

Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.

Include a README?

Yes, with a tutorial (for beginne...)

Include .gitignore?

Yes (recommended)

[Advanced settings](#)

Create repository

Cancel

- Your repository has been created. After that, the following page will appear in front of you.

AOS_Assignment01_m...

Source

Commits

Branches

Pull requests

Pipelines

Jira issues

Downloads

Repository settings

ZARAFSHAN AJMAL / AOS_ASSIGNMENTS

AOS_Assignment01_mscsf19m015

Clone

...

Info

Refresh

Take the next steps for this new repository and its freshly added files

Copy and connect the repository locally so that you can push updates you make and pull changes others make. Enter **git clone** and the repository URL at your command line:

```
git clone https://mscsf19m015@bitbucket.org/mscsf19m015/aos_assignment01_mscsf19m015.git
```

Learn more or clone in Sourcetree to avoid the command line. Sourcetree is a free Git client.

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to your repository.](#)

master

Files

Filter files

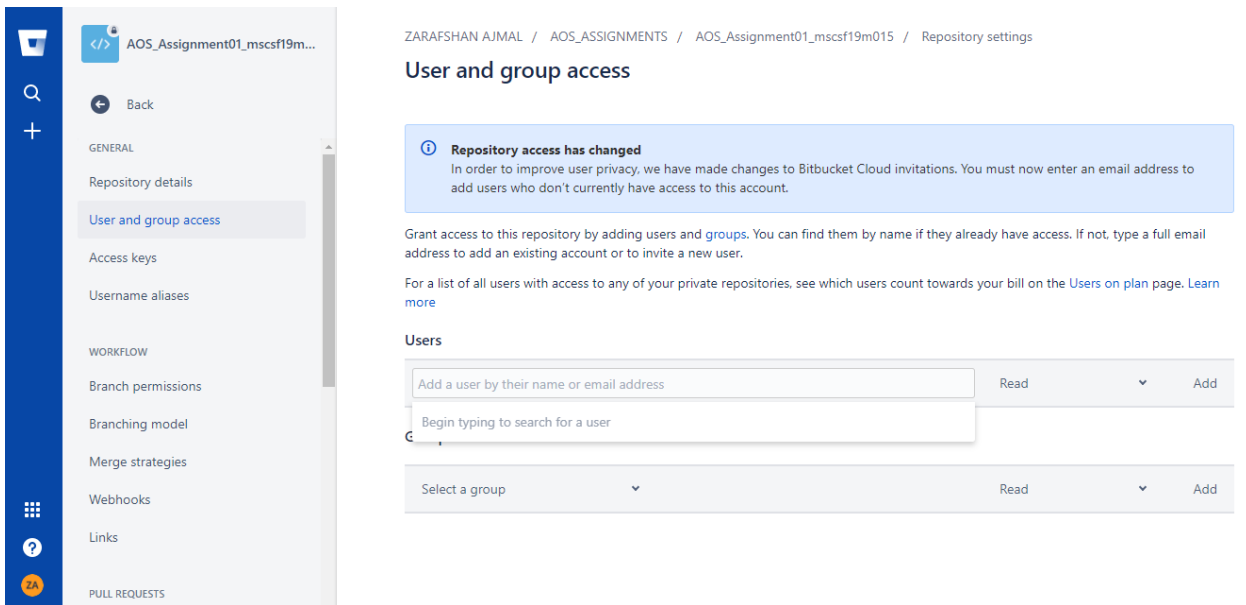
Q

/

Name	Size	Last commit	Message
.gitignore	624 B	22 seconds ago	Initial commit
README.md	2.56 KB	22 seconds ago	Initial commit

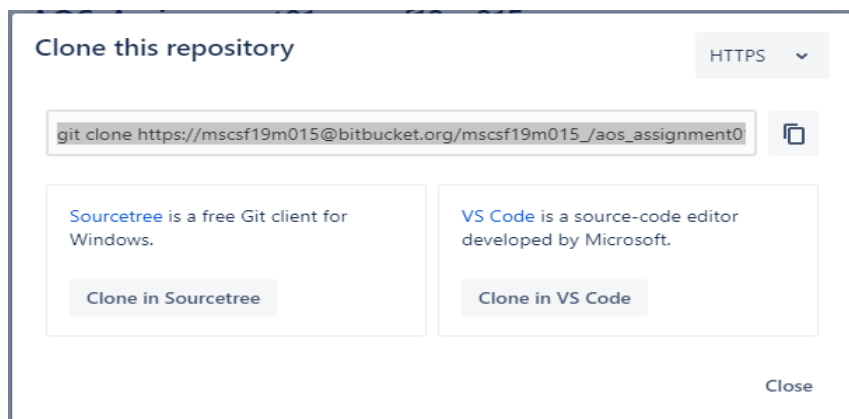
Sharing Repository with Instructor and TAs

- Now you have to make your course instructor (arif@pucit.edu.pk) and course TA (mscsf19m015@pucit.edu.pk) the member of your repository (Access level minimum read). To do this click on the Repository settings on the left hand side and then on the left hand side under General, click **User and Group access**. Here under Users type the email of your course instructor and TAs, and click Add.



Uploading the Assignment folder in the Repository

- Now after creating your repository for the AOS assignment and making course instructor and TA your repository members, we have to attach this online repository with the local repository on your computer so that we can upload the lab folder in the repo. For this purpose first clone or download this repository on your computer. To do this click + on the global sidebar at left and under Get To Work select **Clone this repository**.
- Now copy this link and open a terminal. After opening the terminal window, do the following:
 - Navigate to your home (~) directory.
\$ cd ~
As in future, you can probably work in multiple repositories. For that reason, it's a good idea to create a directory to contain all those repositories.
 - Create a directory to contain your repositories.



\$ mkdir repos

- From the terminal, update the directory you want to work in to your new repos directory.
\$ cd ~/repos
- And now paste the command you copied from Bitbucket.

```

zarafshan@ubuntu:~$
zarafshan@ubuntu:~$
zarafshan@ubuntu:~$ mkdir repos
zarafshan@ubuntu:~$ cd repos
zarafshan@ubuntu:~/repos$ git clone https://mcsf19m015@bitbucket.org/mcsf19m015_/aos_assignment01_mcsf19m015.git
Cloning into 'aos_assignment01_mcsf19m015'...
Password for 'https://mcsf19m015@bitbucket.org':
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
zarafshan@ubuntu:~/repos$ cd aos_assignment01_mcsf19m015/
zarafshan@ubuntu:~/repos/aos_assignment01_mcsf19m015$

```

- Now you have your repo directory on your machine. Now you have to set up your username and password for git. For this purpose, type these commands:

```

$ git config --global user.name "your name"
$ git config --global user.email "your university email"

```

```

zarafshan@ubuntu:~/repos/aos_assignment01_mcsf19m015$
zarafshan@ubuntu:~/repos/aos_assignment01_mcsf19m015$ git config --global user.name "Zarafshan Ajmal"
zarafshan@ubuntu:~/repos/aos_assignment01_mcsf19m015$
zarafshan@ubuntu:~/repos/aos_assignment01_mcsf19m015$ git config --global user.email "mcsf19m015@pucit.edu.pk"

```

- For submission of Assignments, do this:
 - Make a directory having a name, e.g. Assignment01 and go into that directory.
 - Now start doing your Assignment. Make a file for task 1, e.g. L1T1.c and complete this task. Do the same for task 2, 3 and so on.
 - If you have completed the Assignment or done half of Assignment and want to save it on your online repository, then do this:
 - First your file will be sent to the staging area by using git command **git add**, then from staging area it is sent to local repository by using **git commit**, and then from local repository to online or remote repository on bitbucket.
 - Type this in command line to add your files in staging area:

```

$ git add --all (if you want to send all of your files to staging area)
$ git add L1T1.c (if you want to send a specific file to staging area)

```

- You can get the status of your local repository. The **git status** command tells you about how your project is progressing in comparison to your Bitbucket repository.
- Type in the command line to now commit your files and add them into the local repository. It will add all of the files which are in the staging area to the local repository.

```

$ git commit -m "message you want to attach with this commit."

```

- Now to add the files which are in the local repository to the remote repository use this command. This command will add your files to your bitbucket repository.

```

$ git push origin master

```

- Now your Assignment01 files are on your bitbucket repository.

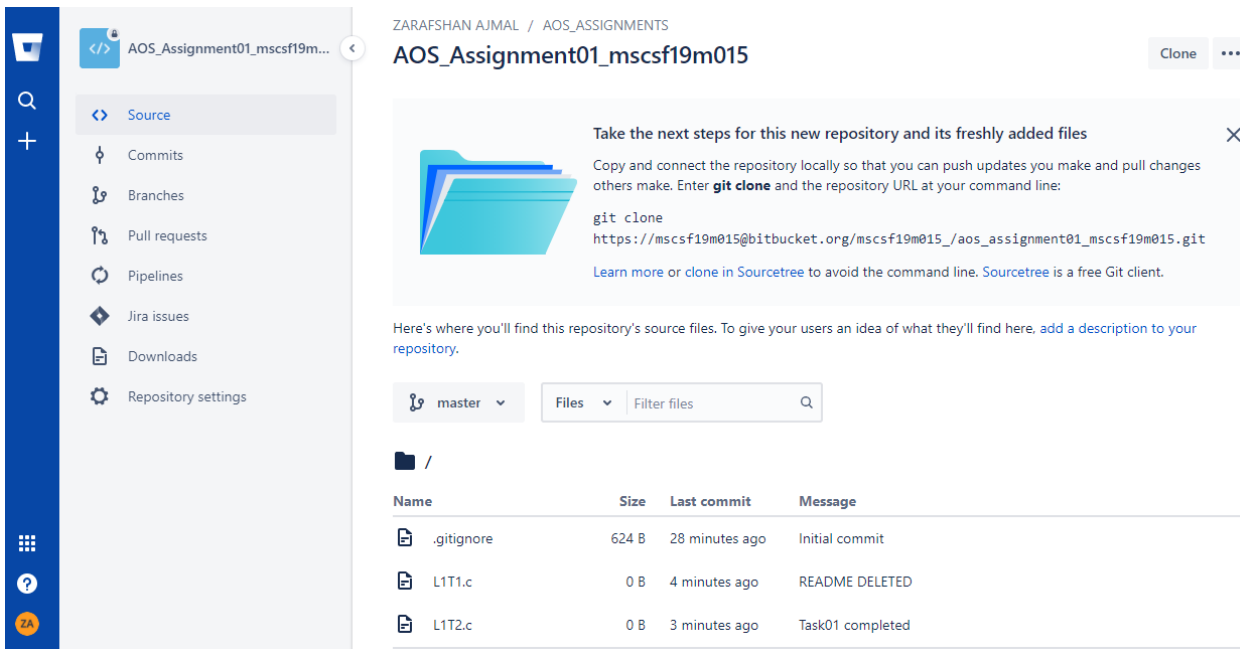
```

zarafshan@ubuntu:~/repos/aos_assignment01_mscsf19m015$
zarafshan@ubuntu:~/repos/aos_assignment01_mscsf19m015$ ls
L1T1.c L1T2.c
zarafshan@ubuntu:~/repos/aos_assignment01_mscsf19m015$ git add --all
zarafshan@ubuntu:~/repos/aos_assignment01_mscsf19m015$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   L1T2.c

zarafshan@ubuntu:~/repos/aos_assignment01_mscsf19m015$ git commit -m "Task01 completed"
[master 87fe12e] Task01 completed
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 L1T2.c
zarafshan@ubuntu:~/repos/aos_assignment01_mscsf19m015$ git push origin master
Password for 'https://mscsf19m015@bitbucket.org':
Counting objects: 5, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 454 bytes | 0 bytes/s, done.
Total 5 (delta 1), reused 0 (delta 0)
To https://mscsf19m015@bitbucket.org/mscsf19m015_/aos_assignment01_mscsf19m015.git
 9013e4b..87fe12e  master -> master
zarafshan@ubuntu:~/repos/aos_assignment01_mscsf19m015$

```



ZARAFSHAN AJMAL / AOS_ASSIGNMENTS

AOS_Assignment01_mscsf19m015

Clone ...

Take the next steps for this new repository and its freshly added files

Copy and connect the repository locally so that you can push updates you make and pull changes others make. Enter **git clone** and the repository URL at your command line:

```
git clone https://mscsf19m015@bitbucket.org/mscsf19m015_/aos_assignment01_mscsf19m015.git
```

Learn more or clone in Sourcetree to avoid the command line. Sourcetree is a free Git client.

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to your repository](#).

master

Name	Size	Last commit	Message
.gitignore	624 B	28 minutes ago	Initial commit
L1T1.c	0 B	4 minutes ago	README DELETED
L1T2.c	0 B	3 minutes ago	Task01 completed

- In short, to submit your Assignment, complete your tasks and just type **git add**, **git commit**, **git push origin master**, and your Assignment will be on your online repository.
- If you are still having problems in lab submission you can get help from this tutorial: <https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud>

Stay Blessed
Happy Learning!