ACM KnowledgeBase Documentation

Release 1.0

Steven Jorgensen, Jacob Fenton

Sep 27, 2017

Contents

Contents:

CHAPTER 1

Makefiles

Contents:

Intro to Makefiles

What is Make?

`make` is a tool that is used to manage dependencies when compiling a program.

An example of this would be if you had 2 source code files `a.inputfile` and `b.inputfile`, and you need to compile `b.inputfile` before `a.inputfile`. The code for a makefile that does this might look something like this:

```
a.output: a.inputfile b.output
    commands to generate a.output from a.inputfile and b.output

b.output: b.inputfile
    commands to generate b.output from b.inputfile
```

If you wanted to build `a.output` all you would need to do is run the command `make a.output`.

Makefile Structure

Makefiles consist of definitions and rules.

A definition would look something like this:

`VAR=value`

When referencing a definition like `VAR`, surround it with `$(VAR)`.

A rule would look something like this:

```
output files: input files
    command to generate outputs from inputs
```

The commands **must** be tab indented to work.

Simple Example Makefile for C

```
CC=gcc

bar: bar.c foo.h foo.o
    $(CC) -o bar bar.c foo.o

foo.o: foo.c foo.h
    $(CC) -c foo.c
```

To run this make file, one would simply run the command `make bar`.

CHAPTER 2

Git Quick Intro and Reference

Introduction

Git is a version control system (VCS) used with programming projects, from individual projects up to large scale enterprise development. Git is extremely useful both for individuals and teams and is an industry standard tool; becoming familiar with it will benefit both your academic career and also make yourself an attractive applicant to companies in industry.

This guide aims to familiarize you with git and to provide a quick reference for your use in the future, as well as to supplement whatever knowledge you may have gained from classes.

So, to kick it off, the big question:

What is git for?

Git, at its lowest level, is for keeping track of versions of a project in multiple locations, allowing you to both share your project across multiple locations, and to keep those locations in sync, while also allowing you to jump back to points in your project history.

Git also provides functionality for keeping different versions of the same project in parallel with 'branches', allowing you to preserve a functional copy of your project while you add features.

This is useful for several reasons:

1. as you make changes to your project, potentially breaking things along the way, you always have access to older, working versions
2. it allows you to keep your projects in sync across multiple computers with more functionality than a tool like Dropbox allows
3. **it makes it easy for your team to collaborate on a project together without stepping on** each-others' toes.
4. **working on the same project across different computers is super easy–when you leave one** computer, you commit and push your changes to the remote server. When you arrive at the other computer, you pull down the changes and resume work.

What is git not for?

1. Storing/versioning binary data. Git, having been designed as a programming tool, works best with plain text data. Its algorithms get very, very confused when handed binary data.
2. Automatic synchronization of files; git is not automatic, you have to tell it what to commit and push and when.

Why should you use git?

1. Backup – you don't have to worry about if your laptop suddenly dies; your data is stored remotely, accessible via git!
2. Manually keeping versions – Say goodbye to `project5_actually_working_no_really-17.cpp`, let git handle keeping track of the different versions of your code!
3. Sharing files with your team is a cakewalk – no more shuffling emails back and forth with zips named `team-project-3_edited_friday_by_jake-5.zip`, or passing around a flash drive, or two people working on it at once and having to manually integrate the changes later. Git can handle all of that!
4. When you make a change at the last minute that breaks your code and you don't know how to fix it; reverting to a working version to hand it is only two commands away!

An Important Note

Git is not GitHub. Git is an open source application, GitHub is a website that provides hosting for projects that use git, as well as many resources used to get started with git. They are very different, and git has no official association with GitHub, beyond the fact that GitHub employs many developers who contribute to the git open source project.

Getting Started

There are a bazillion guides to getting started with git, all of them much better written than anything we could produce. So, to get you started:

an interactive guide to using git on the command line.

a step by step guide to your first GitHub repository.

a guide to using branching in your projects.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search