Debuggers

A debugger gives you control of program execution:

- normal execution (`run`, `cont`)
- stop at a certain point (`break`)
- one statement at a time (`step`, `next`)
- examine program state (`print`)

The gdb debugger

gdb – a line-based interactive debugger.
ddd – an X-windows-based interface for gdb.
To use programs with gdb (or ddd), they must be compiled with the gcc compiler.
gdb (ddd) takes two arguments:

```
% gdb executable
```

E.g.

```
% gdb a.out [code]
```

(The core argument is used if you have a core image because the operating system terminated the program)

gdb sessions

A *session* with gdb is a sequence of commands to control and observe the executable.

```
$ gcc -Wall -g -o prog prog.c
$ gdb prog
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licen
This is free software: you are free to change and redistribute i
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
(gdb) break f
Breakpoint 1 at 0x1082c: file prog.c, line 32.
(gdb) run
```

gdb sessions

```
Starting program: ..../prog
Enter a b c: 1 2 3
Breakpoint 1, f (i=1, j=2) at prog.c:32
32                  a = i + j;
(gdb) next
33                  b = i*i + j*j;
(gdb) next
34                  return a*b;
(gdb) print a
$1 = 3
(gdb) print b
$2 = 5
(gdb) cont
...
```

Basic gdb commands

- `quit` – quits from gdb
- `help [CMD]` – on-line help

Gives information about `CMD` command.

- `run ARGS` – run the program

`ARGS` are whatever you normally use, e.g.

```
% ./prog < data
```

is achieved by:

```
(gdb) run < data
```

gdb status commands

- `where` – stack trace

Find which function the program was executing when it crashed.
Stack may also have references to system error-handling functions.

- `up [N]` – move "up" the stack

Allows you to skip to "scope" of a particular function in stack.

- `list [LINE]` — show code

Displays five lines either side of current statement.

- `print EXPR` – display expression values

`EXPR` may use (current values of) variables.
Special expression `aat1` shows all of the array `a`.

gdb execution commands

- `break [PROC|LINE]` - set break-point

On entry to function `PROC` (or reaching line `LINE`), stop execution and return control to gdb.

- `next` - single step (over functions)

Execute next statement; if statement is a function call, execute entire function body.

- `step` - single step (into functions)

Execute next statement; if statement is a function call, go to first statement in function body.

For more details see gdb's on-line help.

Version Control

Any large, useful software system ...

- will undergo many changes in its lifetime
- multiple programmers making changes
- who may work on the code concurrently and independently

The process of code change needs to be managed so that

- changes produce "consistent" versions of the system
- many programmers can easily work simultaneously
- we can roll back to earlier version if needed
- documentation of when, who, & why changes made
- multiple versions of system can be distributed, tested, merged

Version Control

Consider the following simple scenario:

- a software system contains a source code file `x.c`
- system is worked on by several teams of programmers
- Ann in Sydney adds a new feature in her copy of `x.c`
- Bob in Singapore fixes a bug in his copy of `x.c`

Ultimately, we need to ensure that

- all changes are properly recorded (when, who, why)
- both the new feature and the bug fix are in the next release
- if we later find bugs in old release, they can be fixed

Git

- distributed version control system - multiple repositories, no "master"
- every user has their own repository
- created by Linux Torvalds for Linux kernel
- not better than competitors but better supported/more widely used (e.g. github/bitbucket)
- at first stick with a small subset of commands
- substantial time investment to learn to use Git's full power

Creating a git Repository

- Create repository **git init**
- Copy exiting repository **git clone**

Git uses the sub-directory `.git` to store a local repository.
Inside `.git` is stored all versions of all files under version control (in clever efficient way).
If intereteste dtsartby reading about how git uses SHA-1 hashes.

Tracking a Project with Git

- Project must be in single directory tree.
- Usually don't want to track all files in directory tree
- Don't track binaries, derived files, temporary files, large static files
- Use **.gitignore** files to indicate files never want to track
- Use **git add** *file* to indicate you want to track *file*
- Careful: **git add** *directory* will every file in *file* and sub-directories

Git Commit

- A git commit is a snapshot of all the files in the project.
- Can return the project to this state using **git checkout**
- Beware if you accidentally add a file with confidential info to git - need to remove it from all commits.
- **git add** copies file to staging area for next commit
- **git commit -a** if you want commit current versions of all files being tracked

Git Push/Pull

- git **push** adds commits from your repository to a remote repository
- git **remote** lets you give names to other repositories
- git **pull** adds commits from a remote repository to your repository

Example: making Git Repository Public via Github

Github popular repo hosting site (see competitors e.g. bitbucket)
Github student accounts free for small number of repos
Github and competitors also let you setup collaborators, wiki, web pages, issue tracking
Web access to git repo e.g. https://github.com/mirrors/linux

Example: making Git Repository Public via Github

Its a week after the COMP1511 assignment was due and you want to publish your code to the world.
Create github account - assume you choose *ilove1511* as your login
Create a repository - assume you choose *my_code* for the repo name
Add your ssh key (`.ssh/id_rsa.pub`) to github (Account Settings - SSH Public Keys - Add another public key)

```
cd ~/ass2
git remote add origin git@github.com:ilove1511/my_code.git
git push -u origin master
```