CS143A
Principles on Operating Systems
Discussion 02:

Instructor: Prof. Anton Burtsev

TA: Saehanseul Yi (Hans)

Oct 11, 2019 **2PM**

Agenda

- xv6 installation overview
  - PATH, bashrc (or bash_profile)
  - how the PATH works
- Makefile
- Simple disas example

xv6 Install Overview

```
cd ~
mkdir cs143a
cd cs143a
git clone https://github.com/mit-pdos/6.828-qemu.git qemu

cd qemu
git submodule update --init pixman
./configure --disable-kvm --disable-werror --prefix=/home/<YourUCInetID>/cs143a/qemu-install --target-list=…

make -j 8
make install

export PATH=$PATH:$HOME/cs143a/qemu-install/bin
```

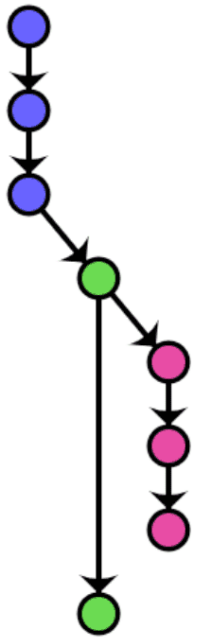
xv6 Install Overview

```
cd ~
mkdir cs143a
cd cs143a
git clone https://github.com/mit-pdos/6.828-qemu.git qemu

cd qemu
git submodule update --init pixman
./configure --disable-kvm --disable-werror --prefix=/home/<YourUCInetID>/cs143a/qemu-install --target-list=…

make -j 8
make install

export PATH=$PATH:$HOME/cs143a/qemu-install/bin
```

Git: Version control system

git log

```
commit d531b1b1d6b7696dfd9695c1d560e3df53e615c5
Author: Lef Ioannidis <eleftheI@mit.edu>
Date:   Thu Sep 6 22:11:54 2018 -0400

    Apply OSX fixes, test first on linux

commit 1f73b5e0fbe1b27f1a5d3f2e1aeb2253e2529c29
Author: Cody Cutler <ccutler@csail.mit.edu>
Date:   Wed Sep 7 07:16:23 2016 -0400

    fix build

commit 0227b1dd681b0975cfeb18b001dad25a17a94d53
Author: Xi Wang <xi@cs.washington.edu>
Date:   Fri Oct 2 20:15:51 2015 -0700

    remove g_mem_set_vtable()

    This should avoid the warning:

    GLib-WARNING **: gmem.c:482: custom memory allocation vtable not supported
```



2e66eaa This is a commit - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

084b579 And here's another one! - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

4a59a19 He doesn't like George Michael! Boooo! - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

ed31895 My first commit on develop. - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

d251c1e This is a mandatory feature - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

176986d He doesn't like George Michael! Boooo! - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

0808818 He doesn't like George Michael! Boooo! - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

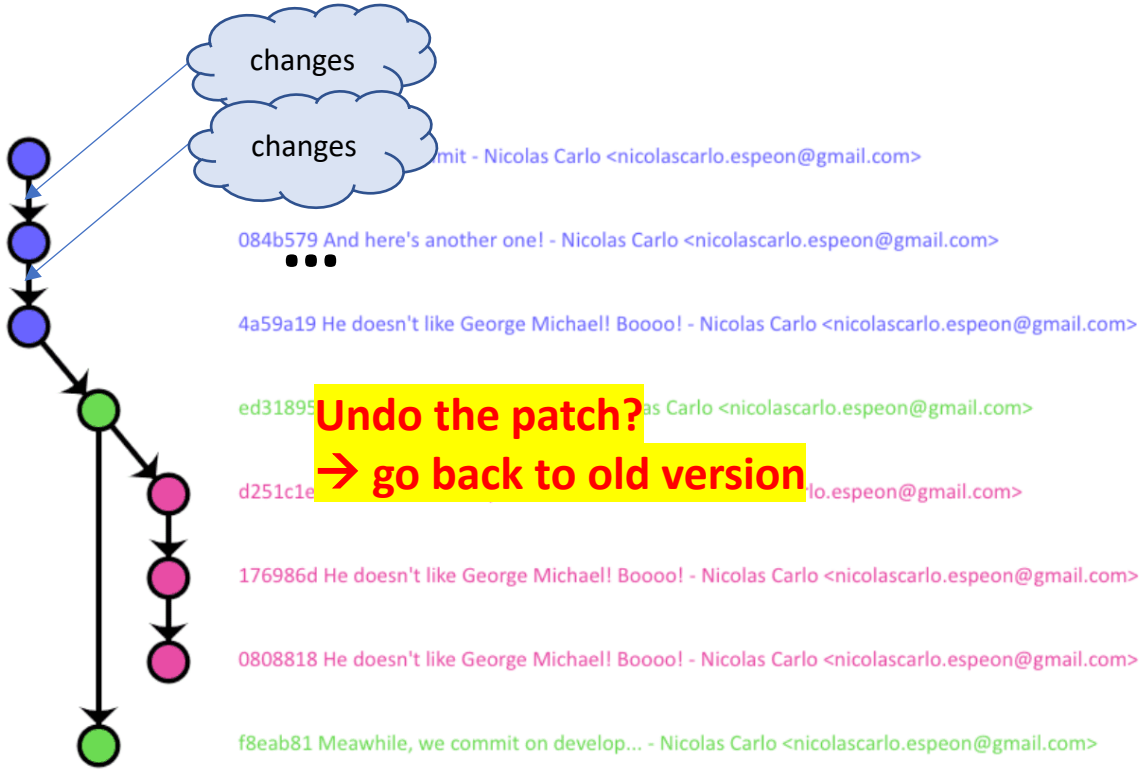
f8eab81 Meawhile, we commit on develop... - Nicolas Carlo <nicolascarlo.espeon@gmail.com>

picture from https://www.nicoespeon.com/projects

Git: Version control system

git log

```
commit d531b1b1d6b7696dfd9695c1d560e3df53e615c5
Author: Lef Ioannidis <elefthei@mit.edu>
Date:   Thu Sep 6 22:11:54 2018 -0400

    Apply OSX fixes, test first on linux

commit 1f73b5e0fbe1b27f1a5d3f2e1aeb2253e2529c29
Author: Cody Cutler <ccutler@csail.mit.edu>
Date:   Wed Sep 7 07:16:23 2016 -0400

    fix build

commit 0227b1dd681b0975cfeb18b001dad25a17a94d53
Author: Xi Wang <xi@cs.washington.edu>
Date:   Fri Oct 2 20:15:51 2015 -0700

    remove g_mem_set_vtable()

    This should avoid the warning:

    GLib-WARNING **: gmem.c:482: custom memory allocation vtable not supported
```



commit: code changes(patch)

picture from https://www.nicoespeon.com/projects

Git : Version control system

git status (list the modified files)

```
saehansy@circinus-30 09:15:32 ~/Workspace/ics143a/FQ19/qemu
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#   (commit or discard the untracked or modified content in submodules)
#
#       modified:   pixman (untracked content)
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       qemu-install/
#       xv6-public/
no changes added to commit (use "git add" and/or "git commit -a")
```

git diff (shows the patch)

```
saehansy@circinus-30 09:08:47 ~/Workspace/ics143a/FQ19/qemu
$ git diff
diff --git a/pixman b/pixman
--- a/pixman
+++ b/pixman
@@ -1 +1 @@
-Subproject commit 87eea99e443b389c978cf37efc52788bf03a0ee0
+Subproject commit 87eea99e443b389c978cf37efc52788bf03a0ee0-dirty
```

24 xcs_classifier_system.cpp

```
@@ -990,13 +990,8 @@ xcs_classifier_system::genetic_algorithm(t_classifier_set &action_set, const t_s
990 990   }
991 991
992 992   void
993     - xcs_classifier_system::step(const bool exploration_mode, const bool condensationMode)
    993 + xcs_classifier_system::step_part1(const bool exploration_mode, const bool condensationMode, t_action& ac
994 994   {
995     -     t_action  action;                                   //! selected action
996     -     //unsigned long   action_set_size;        //! number of microclassifiers in [A]
997     -     double            P;                                          //! value for prediction update, c
998     -     double            max_prediction;
999     -
1000 995       //! reads the current input
1001 996       current_input = Environment->state();
1002 997
```

More info:
Pro Git (free ebook)
https://git-scm.com/book/en/v2

from github..

xv6 Install Overview

```
cd ~
mkdir cs143a
cd cs143a
git clone https://github.com/mit-pdos/6.828-qemu.git qemu

cd qemu
git submodule update --init pixman
./configure --disable-kvm --disable-werror --prefix=/home/<YourUCInetID>/cs143a/qemu-install --target-list=…

make -j 8
make install

export PATH=$PATH:$HOME/cs143a/qemu-install/bin
```

a subproject called “pixman” inside this git repository

xv6 Install Overview

```
cd ~
mkdir cs143a
cd cs143a
git clone https://github.com/mit-pdos/6.828-qemu.git qemu

cd qemu
git submodule update --init pixman
./configure --disable-kvm --disable-werror --prefix=/home/<YourUCInetID>/cs143a/qemu-install --target-list=…

make -j 8
make install

export PATH=$PATH:$HOME/cs143a/qemu-install/bin
```

When building a program from source code…

- **./configure**: getting ready to build the software on your specific system
- **make**: build the software in the directory where the source code is
- **make install**: copy the software(binary, library, documentations..) to correct locations

Correct locations?
(default)
For binaries, /usr/bin
For libraries, /usr/lib
…
or it can be set by **--prefix**.
But you should add your custom path to PATH

For libraries, LD_LIBRARY_PATH

When you type a command, the system searches the software in $PATH

```
saehansy@circinus-30 09:15:44 ~/Workspace/ics143a/FQ19/qemu
$ echo $PATH
/home/saehansy/Workspace/ics143a/FQ19/qemu/qemu-install/bin:/home/saehansy/Works
pace/ics143a/qemu-install/bin:/home/saehansy/local/bin:/pkg/gsu/3.0b/bin:/usr/bi
n:/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/
opt/puppetlabs/bin
```

When building a program from source code…

executable

library(shared object): needed by driver

```
saehansy@circinus-30 10:20:58 ~/Workspace/PIC
$ ls
Makefile  driver*  driver.c  driver.o  hex_libmlreloc.so.txt  libmlreloc.so*  ml_main.c  ml_mainreloc.o
 nm.txt  objdump.txt  readelf_h.txt  readelf_r.txt  readelf_segments.txt  tags  typescript
```

```
saehansy@circinus-30 10:24:01 ~/Workspace/PIC
$ gdb driver -q
Reading symbols from /home/saehansy/Workspace/PIC/driver...done.
(gdb) b 28
Breakpoint 1 at 0x804863e: file driver.c, line 28.
(gdb) run
Starting program: /home/saehansy/Workspace/PIC/driver
/home/saehansy/Workspace/PIC/driver: error while loading shared libraries: libmlreloc.so:
cannot open shared object file: No such file or directory
[Inferior 1 (process 21410) exited with code 0177]
Missing separate debuginfos, use: debuginfo-install glibc-2.17-292.el7.i686
(gdb) quit
saehansy@circinus-30 10:24:27 ~/Workspace/PIC
```

```
saehansy@circinus-30 10:27:31 ~/Workspace/PIC
$ export LD_LIBRARY_PATH=$HOME/Workspace/PIC:$LD_LIBRARY_PATH
```

```
saehansy@circinus-30 10:27:38 ~/Workspace/PIC
$ gdb -q driver
Reading symbols from /home/saehansy/Workspace/PIC/driver...done.
(gdb) b 28
Breakpoint 1 at 0x804863e: file driver.c, line 28.
(gdb) run
Starting program: /home/saehansy/Workspace/PIC/driver

Breakpoint 1, main (argc=1, argv=0xffffce04) at driver.c:28
28          dl_iterate_phdr(header_handler, NULL);
Missing separate debuginfos, use: debuginfo-install glibc-2.17-292.el7.i686
(gdb) disas
Dump of assembler code for function main:
   0x08048635 <+0>:     push   %ebp
   0x08048636 <+1>:     mov    %esp,%ebp
   0x08048638 <+3>:     and    $0xfffffff0,%esp
   0x0804863b <+6>:     sub    $0x20,%esp
=> 0x0804863e <+9>:     movl   $0x0,0x4(%esp)
   0x08048646 <+17>:    movl   $0x804856d,(%esp)
   0x0804864d <+24>:    call   0x8048430 <dl_iterate_phdr@plt>
   0x08048652 <+29>:    mov    0x8(%ebp),%eax
   0x08048655 <+32>:    mov    %eax,0x4(%esp)
   0x08048659 <+36>:    mov    0x8(%ebp),%eax
   0x0804865c <+39>:    mov    %eax,(%esp)
   0x0804865f <+42>:    call   0x8048420 <ml_func@plt>
   0x08048664 <+47>:    mov    %eax,0x1c(%esp)
   0x08048668 <+51>:    mov    0x1c(%esp),%eax
   0x0804866c <+55>:    leave
   0x0804866d <+56>:    ret
End of assembler dump.
(gdb) quit
```

When building a program from source code…

- **export PATH=$PATH:$HOME/cs143a/qemu-install/bin**
  - NOTE: there's no $ when you set variable
  - $ is for reading the value of the variable
  - : to append another path
- Checking the variable values
  - echo $HOME
  - echo $PATH
- Whenever you are log in to terminal, it executes .bashrc
  - Add necessary commands to .bashrc

xv6 Install Overview

```
cd ~
mkdir cs143a
cd cs143a
git clone https://github.com/mit-pdos/6.828-qemu.git qemu

cd qemu
git submodule update --init pixman
./configure --disable-kvm --disable-werror --prefix=/home/<YourUCInetID>/cs143a/qemu-install --target-list=…

make -j 8
make install

export PATH=$PATH:$HOME/cs143a/qemu-install/bin
```

j is an option for multi-threaded compilation (e.g. use 8 threads)

Makefile

target: prerequisites
<TAB> recipe

```
all:
    gcc -c -g main.c -o a.exe
```

Using variables

```
CC=gcc
all:
    $(CC) -c -g main.c -o a.exe
```

Printing variables

```
CC=gcc
all:
    $(info CC is ${CC})
    $(CC) -c -g main.c -o a.exe
```

→ cc is gcc

Commenting out(#)

```
1 all:
2         gcc -c -g -m32 -fno-pic ml_main.c -o ml_mainreloc.o
3         gcc -m32 -shared -o libmlreloc.so ml_mainreloc.o
4         gcc -g -c -m32 driver.c -o driver.o
5         gcc -m32 -o driver driver.o -L. -lmlreloc
6         #gcc -shared -c -g -m32 -fno-pic ml_main.c -o ml_mainreloc.so
```

Objdump

- dump object file(including executables) information
  - useful for static analysis, no need to run gdb
- Linux redirection (>)
  - In default, outputs will be printed out to stdout(which is your terminal screen)
  - Redirect this to somewhere else (e.g. a text file)
  - objdump -d a.out > objdump.txt
- objdump -S(source code) or objdump -D(disassemble)
- Tip: Hex viewer in VIM :%!xxd

```
$ objdump -S driver

driver:     file format elf32-i386


Disassembly of section .init:

080483dc <_init>:
 80483dc:       53                      push   %ebx
 80483dd:       83 ec 08                sub    $0x8,%esp
 80483e0:       e8 bb 00 00 00          call   80484a0 <__x86.get_pc_thunk.bx>
 80483e5:       81 c3 1b 1c 00 00       add    $0x1c1b,%ebx
 80483eb:       8b 83 fc ff ff ff       mov    -0x4(%ebx),%eax
 80483f1:       85 c0                   test   %eax,%eax
 80483f3:       74 05                   je     80483fa <_init+0x1e>
 80483f5:       e8 66 00 00 00          call   8048460 <.plt.got>
 80483fa:       83 c4 08                add    $0x8,%esp
 80483fd:       5b                      pop    %ebx
 80483fe:       c3                      ret
```

Simple *disas* example

- https://www.ics.uci.edu/~aburtsev/143A/hw/hw1-simple-programs.html
- Download main.c
- gcc main.c -o hello

GNU Debugger(GDB)

- Control the execution flow of the program (stop/resume)
- View/modify the system status (register, memory contents, …)
- Run the target(inferior) inside gdb or *attach* to the running process
- Even remote debugging is possible (through network)

GNU Debugger(GDB)

- Check debug information
  - l (or list)

```
list
list <filename>:<function>
list <filename>:<line_number>
```

```
(gdb) l
1       #include <stdio.h>
2
3       int main()
4       {
5       char str[2][3] = {0,};
6       printf ("%p\n", str);
7       printf ("%p\n", &str[0]);
8       printf ("%p\n", &str[1]);
9       printf ("%p\n", &str[1][0]);
10      printf ("%p\n", &str[2]);
(gdb) list
11      printf ("%p\n", &str[2][0]);
12      printf ("%p\n", &str[2][1]);
13      return 0;
14      }
15
16
```

GNU Debugger(GDB)

- breakpoint: stop the program at certain point
  - where?
    - a line of the source code
    - or at specific memory address
- info b: list breakpoints
- delete <num>

```
(gdb) break 5
Breakpoint 1 at 0x400525: file test.c, line 5.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000400525 in main at test.c:5
(gdb) delete 1
(gdb) info b
No breakpoints or watchpoints.
(gdb) break 5
Breakpoint 2 at 0x400525: file test.c, line 5.
(gdb) run
Starting program: /home/saehansy/Workspace/ics143a/FQ19/test.exe

Breakpoint 2, main () at test.c:5
5       char str[2][3] = {0,};
```

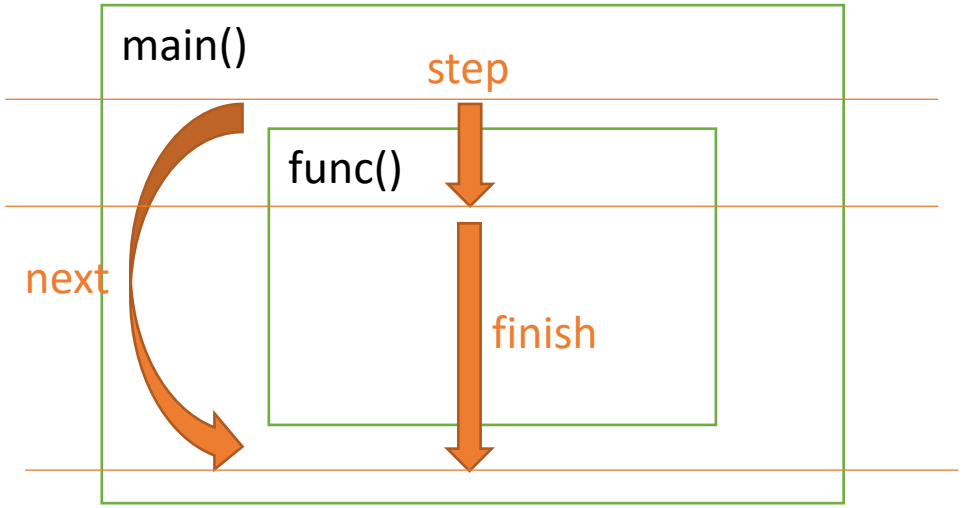
GNU Debugger(GDB)

- run & continue
  - **run**: run the program. If there's no breakpoint, the program will run until the end as if there is no gdb
  - **continue**: when program stopped at some breakpoint, *continue* will make the program run until the next breakpoint; otherwise, no further breakpoint, it run until the end

GNU Debugger(GDB)

- next, step in & out
  - step over: execute one line (gdb command: next)
  - step in: execute one line & go inside the function (gdb command: step)
  - step out: skip the rest of the current function (gdb command: finish)

```
(gdb) step
step      stepi     stepping
(gdb) stepi
0x000000000040052c      5       char str[2][3] = {0,};
(gdb)
6       printf ("%p\n", str);
(gdb)
0x0000000000400536      6       printf ("%p\n", str);
(gdb)
0x0000000000400539      6       printf ("%p\n", str);
```

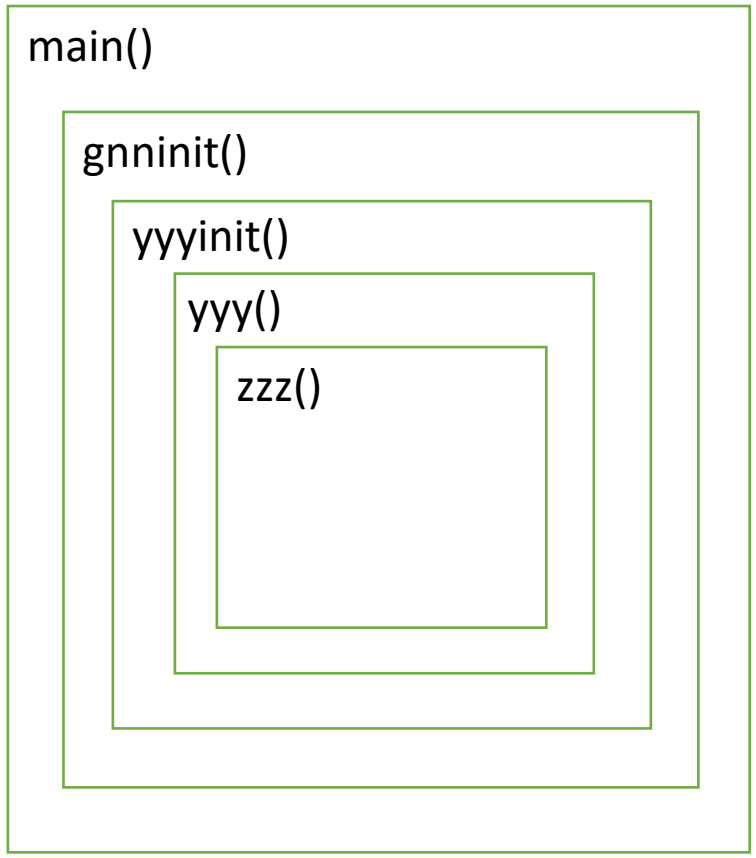


- execute one instruction: stepi, nexti

GNU Debugger(GDB)

- bt (or backtrace): shows the *call stack*

```
(gdb) bt
#0 zzz () at zzz.c:96
#1 0xf7d39cba in yyy (arg=arg@entry=0x0) at
yyy.c:542
#2 0xf7d3a4f6 in yyyinit () at yyy.c:590
#3 0x0804ac0c in gnninit () at gnn.c:374
#4 main (argc=1, argv=0xffffd5e4) at gnn.c:389
```



GNU Debugger(GDB)

- info & help
  - **info reg**
  - **info frame**

```
(gdb) info reg
rax            0x7fffffffdbe0   140737488346080
rbx            0x0      0
rcx            0x4005f0 4195824
rdx            0x7fffffffdce8   140737488346344
rsi            0x7fffffffdbe0   140737488346080
rdi            0x1      1
rbp            0x7fffffffdbf0   0x7fffffffdbf0
rsp            0x7fffffffdbe0   0x7fffffffdbe0
r8             0x7ffff7dd5e80   140737351868032
r9             0x0      0
r10            0x7fffffffd880   140737488345216
r11            0x7ffff7a302e0   140737348043488
r12            0x400430 4195376
r13            0x7fffffffdcd0   140737488346320
r14            0x0      0
r15            0x0      0
rip            0x400539 0x400539 <main+28>
eflags         0x202    [ IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
```

```
(gdb) info
address                 copying                 inferiors
all-registers           dcache                  line
args                    display                 locals
auto-load               extensions              macro
auxv                    files                   macros
bookmarks               float                   mem
breakpoints             frame                   os
checkpoints             frame-filter            pretty-printer
classes                 functions               probes
common                  handle                  proc
```

```
(gdb) help stepping
Specify single-stepping behavior at a tracepoint.
Argument is number of instructions to trace in single-step mode
following the tracepoint.  This command is normally followed by
one or more "collect" commands, to specify what to collect
while single-stepping.
```

```
(gdb) info frame
Stack level 0, frame at 0x7fffffffdc00:
 rip = 0x400539 in main (test.c:6); saved rip 0x7ffff7a303d5
 source language c.
 Arglist at 0x7fffffffdbf0, args:
 Locals at 0x7fffffffdbf0, Previous frame's sp is 0x7fffffffdc00
 Saved registers:
  rbp at 0x7fffffffdbf0, rip at 0x7fffffffdbf8
```

GNU Debugger(GDB)

- Debugging assembly
  - **objdump –D <exec>:** human-readable dump of instructions of a program
- Additional windows(helpful)
  - In some systems, **tui enable** – **layout asm** – **tui disable**
  - or **tui reg general** – **layout asm**
  - To turn it off, C-x a(or C-x C-a, no need to lift the control key up)

```
Register group: general
rax            0x7fffffffdbe0   140737488346080        rbx            0x0      0                              rcx            0x4005f0 4195824
rdx            0x7fffffffdce8   140737488346344        rsi            0x7fffffffdbe0   140737488346080       rdi            0x1      1
rbp            0x7fffffffdbf0   0x7fffffffdbf0         rsp            0x7fffffffdbe0   0x7fffffffdbe0        r8             0x7ffff7dd5e80   140737351868032
r9             0x0      0                              r10            0x7fffffffd880   140737488345216       r11            0x7ffff7a302e0   140737348043488
r12            0x400430 4195376                        r13            0x7fffffffdcd0   140737488346320       r14            0x0      0
r15            0x0      0                              rip            0x400539 0x400539 <main+28>            eflags         0x202    [ IF ]
cs             0x33     51                             ss             0x2b     43                            ds             0x0      0
es             0x0      0                              fs             0x0      0                             gs             0x0      0
```

Window name: *regs*

test.c

```
2
3       int main()
4       {
B+ 5    char str[2][3] = {0,};
  > 6   printf ("%p\n", str);
7       printf ("%p\n", &str[0]);
8       printf ("%p\n", &str[1]);
9       printf ("%p\n", &str[1][0]);
10      printf ("%p\n", &str[2]);
11      printf ("%p\n", &str[2][0]);
12      printf ("%p\n", &str[2][1]);
13      return 0;
14      }
15
```

Window name: *src*

```
child process 24680 In: main                                                    Line: 6    PC: 0x400539
(gdb)
```

```
> 0x400539 <main+28>      mov    $0x400680,%edi
  0x40053e <main+33>      mov    $0x0,%eax
  0x400543 <main+38>      callq  0x400400 <printf@plt>
  0x400548 <main+43>      lea    -0x10(%rbp),%rax
  0x40054c <main+47>      mov    %rax,%rsi
  0x40054f <main+50>      mov    $0x400680,%edi
  0x400554 <main+55>      mov    $0x0,%eax
  0x400559 <main+60>      callq  0x400400 <printf@plt>
  0x40055e <main+65>      lea    -0x10(%rbp),%rax
  0x400562 <main+69>      add    $0x3,%rax
  0x400566 <main+73>      mov    %rax,%rsi
  0x400569 <main+76>      mov    $0x400680,%edi
  0x40056e <main+81>      mov    $0x0,%eax
  0x400573 <main+86>      callq  0x400400 <printf@plt>
  0x400578 <main+91>      lea    -0x10(%rbp),%rax
  0x40057c <main+95>      add    $0x3,%rax
  0x400580 <main+99>      mov    %rax,%rsi
  0x400583 <main+102>     mov    $0x400680,%edi
  0x400588 <main+107>     mov    $0x0,%eax
  0x40058d <main+112>     callq  0x400400 <printf@plt>
  0x400592 <main+117>     lea    -0x10(%rbp),%rax
  0x400596 <main+121>     add    $0x6,%rax
  0x40059a <main+125>     mov    %rax,%rsi
  0x40059d <main+128>     mov    $0x400680,%edi
  0x4005a2 <main+133>     mov    $0x0,%eax
  0x4005a7 <main+138>     callq  0x400400 <printf@plt>
  0x4005ac <main+143>     lea    -0x10(%rbp),%rax
  0x4005b0 <main+147>     add    $0x6,%rax
  0x4005b4 <main+151>     mov    %rax,%rsi
```

Window name: *asm*

```
child process 24680 In: main                                    Line: 6    PC: 0x400539
(gdb)
```

GNU Debugger(GDB)

- Adjust window height
  - winheight <name> +*count*
  - winheight <name> -*count*
  - <name>: src, asm, regs, and cmd
- ***refresh***: sometimes the window layout breaks (e.g. if you +/- font size, adjust the terminal window, …). Then, ‘refresh’ refreshes the screen

https://sourceware.org/gdb/onlinedocs/gdb/TUI-Commands.html

GNU Debugger(GDB)

- breakpoints using address
  - b *0x4005b4
  - For addresses, use * in front of it
- Useful print command
  - **p (or print)** <var_name> or *<address> or $registers
  - **x/[NUM][FMT] $sp**: show stack memory; FMT can be x(hex) f(float), …

prints 10 words in hexadecimal above the stack pointer($sp)

```
(gdb) x/10x $sp
0xffeac63c: 0xf7d39cba 0xf7d3c0d8 0xf7d3c21b 0x00000001
0xffeac64c: 0xf78d133f 0xffeac6f4 0xf7a14450 0xffeac678
0xffeac65c: 0x00000000 0xf7d3790e
```

GNU Debugger(GDB)

- For more information, search for “GDB cheatsheet”
  - https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf