

Operating Systems Practice

Debugging with GDB

Juhyung Park

arter97@dgist.ac.kr

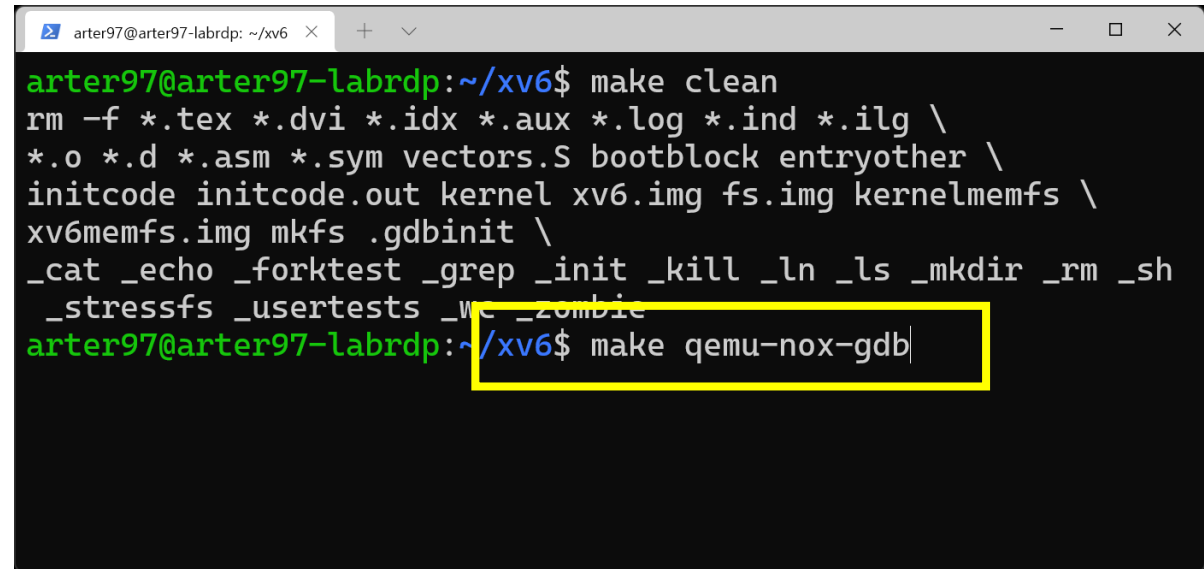


Xv6 + GDB

- Debugging xv6 with GDB

Terminal 1: Run gdb to send a command

Terminal 2: Run xv6 with gdb



```
arter97@arter97-labrdp: ~/xv6 $ make clean
rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
*.o *.d *.asm *.sym vectors.S bootblock entryother \
initcode initcode.out kernel xv6.img fs.img kernelmemfs \
xv6memfs.img mkfs .gdbinit \
_cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh
_stressfs _usertests _wc _zombie
arter97@arter97-labrdp: ~/xv6 $ make qemu-nox-gdb|
```

Xv6 + GDB

- Debugging xv6 with GDB

Terminal 1: Run gdb to send a command

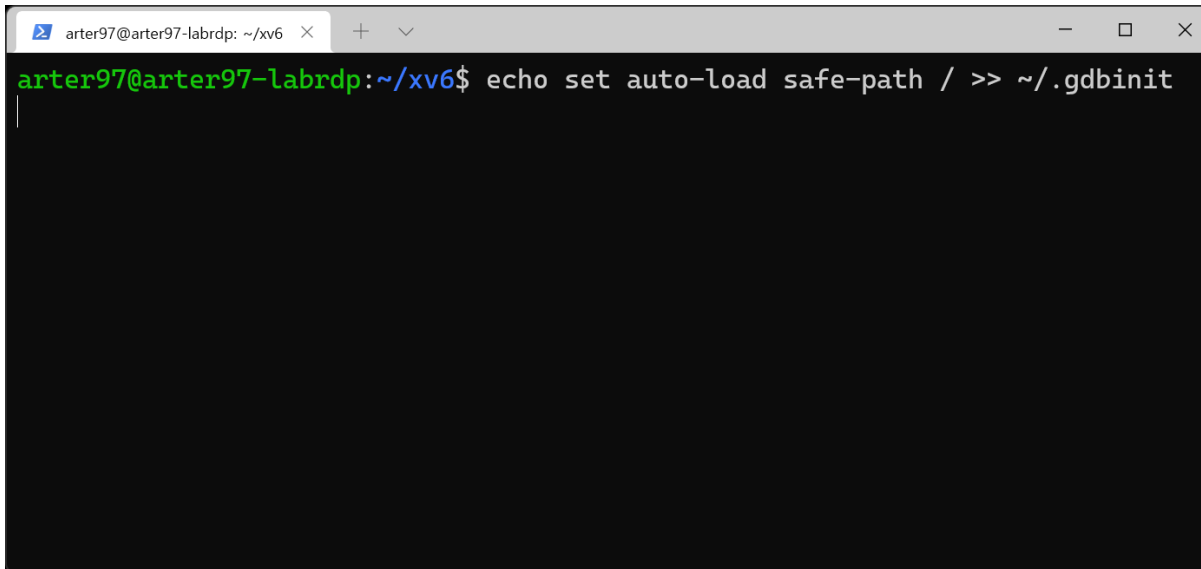
Terminal 2: Run xv6 with gdb

```
arter97@arter97-labrdp: ~/xv6  ×  +  -  □  ×  
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0231575 s, 221 MB/s  
dd if=bootblock of=xv6.img conv=notrunc  
1+0 records in  
1+0 records out  
512 bytes copied, 0.0001367 s, 3.7 MB/s  
dd if=kernel of=xv6.img seek=1 conv=notrunc  
416+1 records in  
416+1 records out  
213276 bytes (213 kB, 208 KiB) copied, 0.000751 s, 284 MB/s  
sed "s/localhost:1234/localhost:26000/" < .gdbinit.tmpl > .gt  
*** Now run 'gdb'.  
qemu-system-i386 -nographic -drive file=fs.img,index=1,media0
```


Xv6 + GDB

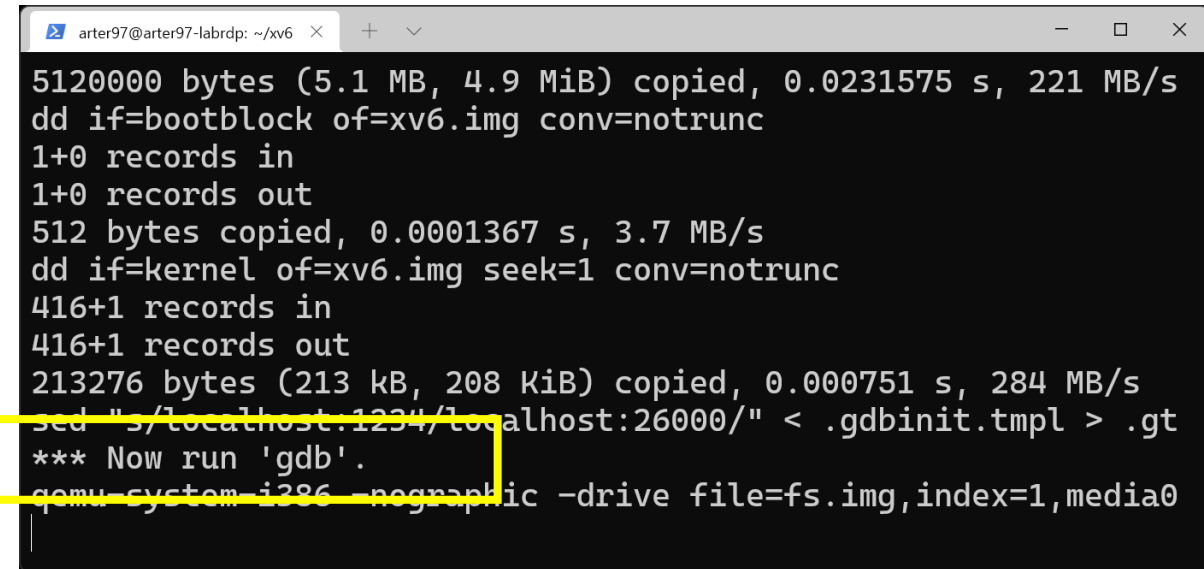
- Debugging xv6 with GDB

Terminal 1: Run gdb to send a command

A terminal window with a dark background and light green text. The prompt is 'arter97@arter97-labrdp: ~/xv6'. The command entered is 'echo set auto-load safe-path / >> ~/.gdbinit'.

```
arter97@arter97-labrdp: ~/xv6$ echo set auto-load safe-path / >> ~/.gdbinit
```

Terminal 2: Run xv6 with gdb

A terminal window with a dark background and light green text. It shows the output of several 'dd' commands copying bootblock, kernel, and fs.img to xv6.img. The last command is 'sed "s/localhost:1234/localhost:26000/" < .gdbinit.tmpl > .gt', which is highlighted with a yellow box. Below it is the command 'qemu-system-i386 -nographic -drive file=fs.img,index=1,media0'.

```
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0231575 s, 221 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.0001367 s, 3.7 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
416+1 records in
416+1 records out
213276 bytes (213 kB, 208 KiB) copied, 0.000751 s, 284 MB/s
sed "s/localhost:1234/localhost:26000/" < .gdbinit.tmpl > .gt
*** Now run 'gdb'.
qemu-system-i386 -nographic -drive file=fs.img,index=1,media0
```

One-time setup

```
echo set auto-load safe-path / >> ~/.gdbinit
```

Xv6 + GDB

- Debugging xv6 with GDB

Terminal 1: Run gdb to send a command

```
arter97@arter97-labrdp: ~/xv6 $ gdb kernel
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.ht
ml>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel...
+ target remote localhost:26000
The target architecture is assumed to be i386
[f000:fff0] 0xfffff0: jmp $0x3630,$0xf000e05b
0x0000fff0 in ?? ()
+ symbol-file kernel
(gdb) |
```

Terminal 2: Run xv6 with gdb

```
arter97@arter97-labrdp: ~/xv6 $ dd if=bootblock of=xv6.img conv=notrunc
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0231575 s, 221 MB/s
1+0 records in
1+0 records out
512 bytes copied, 0.0001367 s, 3.7 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
416+1 records in
416+1 records out
213276 bytes (213 kB, 208 KiB) copied, 0.000751 s, 284 MB/s
sed "s/localhost.1234/localhost:26000/" < .gdbinit.tpl > .gt
*** Now run 'gdb'.
qemu-system-i386 -nographic -drive file=fs.img,index=1,media0
```

Xv6 + GDB

- Debugging xv6 with GDB

Terminal 1: Run gdb to send a command

```
arter97@arter97-labrdp: ~/xv6
+ target remote localhost:26000
The target architecture is assumed to be i8086
[f000:fff0] 0xfffff0: ljmp $0x3630,$0xf000e05b
0x0000fff0 in ?? ()
+ symbol-file kernel
(gdb) b exec
Breakpoint 1 at 0x80100a80: file exec.c, line 12.
(gdb) c
Continuing.
The target architecture is assumed to be i386
=> 0x80100a80 <exec>: endbr32

Thread 1 hit Breakpoint 1, exec (path=0x1c "/init", argv=0x8dffffed0)
at exec.c:12
12 {
(gdb) |
```

Terminal 2: Run xv6 with gdb

```
arter97@arter97-labrdp: ~/xv6
SeaBIOS (version 1.13.0-1ubuntu1.1)

iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA100

Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 ino8
|
```

Xv6 + GDB

- Debugging xv6 with GDB

Terminal 1: Run gdb to send a command

```
arter97@arter97-labrdp: ~/xv6
0x801028a9 in lapicid () at lapic.c:103
103     if (!lapic)
(gdb) n
=> 0x801028ad <lapicid+13>:      mov     0x20(%eax),%eax
105     return lapic[ID] >> 24;
(gdb) n

Thread 1 received signal SIGTRAP, Trace/breakpoint trap.
[Switching to Thread 1.1]
=> 0x801038ed <mycpu+29>:      test    %esi,%esi
mycpu () at proc.c:48
48     for (i = 0; i < ncpu; ++i) {
1: apicid = 0
(gdb) display ncpu
2: ncpu = 2
(gdb) |
```

Terminal 2: Run xv6 with gdb

```
arter97@arter97-labrdp: ~/xv6
SeaBIOS (version 1.13.0-1ubuntu1.1)

iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA100

Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 ino8
|
```



Commands

- Run or Exit the program
 - c (continue) : start/continue the program
 - n (next) : execute next line
 - s (step): execute next line (if next line calls functions, move inside the function)
 - ctrl + c : interrupt/break xv6 (useful on endless loops)
 - q (quit) : exit gdb

Commands

- Break points
 - b filename.c:num : Set break point on *num*'th line on filename.c
 - b *num* : Set break point on *num* line
 - b *num* if 'condition' : If condition is satisfied during program execution, break *num* line
 - e.g., b 10 if var==20
 - i b (info break) : print current break points and watch points
 - d *num* (delete): Release breakpoint on *num* break point
 - watch *var* : Program breaks whenever the value of *var* variable changes



Commands

- Print variables
 - `p var` : print current value of *var* variable
 - `p *ptr` : Output the value referenced by the variable when the *ptr* variable is a pointer
 - `display var` : Print the *var* variable on every GDB interrupt
 - `info locals/variables` : list "Local variables of current stack frame"/"All global and static variable names".
- Trace (backtrace)
 - `bt` : summary of how your program got where it is.
 - `bt full` : Print the values of the local variables also.