[yjarrar@linux xv6-public]$ gdb

GNU gdb (GDB) Fedora 8.2-5.fc29

Copyright (C) 2018 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

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-redhat-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".

warning: File "/home/yjarrar/Documents/Lab 1/xv6-public/.gdbinit" auto-loading has been declined by your `auto-load safe-path' set to "$debugdir:$datadir/auto-load".

To enable execution of this file add

add-auto-load-safe-path /home/yjarrar/Documents/Lab 1/xv6-public/.gdbinit

line to your configuration file "/home/yjarrar/.gdbinit".

To completely disable this security protection add

set auto-load safe-path /

line to your configuration file "/home/yjarrar/.gdbinit".

For more information about this security protection see the

--Type <RET> for more, q to quit, c to continue without paging--c

"Auto-loading safe path" section in the GDB manual. E.g., run from the shell:

info "(gdb)Auto-loading safe path"

(gdb) target remote:26000

Remote debugging using :26000

warning: Remote gdbserver does not support determining executable automatically.

RHEL <=6.8 and <=7.2 versions of gdbserver do not support such automatic executable detection.

The following versions of gdbserver support it:

- Upstream version of gdbserver (unsupported) 7.10 or later

- Red Hat Developer Toolset (DTS) version of gdbserver from DTS 4.0 or later (only on x86\_64)

- RHEL-7.3 versions of gdbserver (on any architecture)

warning: No executable has been specified and target does not support

determining executable automatically. Try using the "file" command.

0x0000fff0 in ?? ()

(gdb) file kernel

A program is being debugged already.

Are you sure you want to change the file? (y or n) y

Reading symbols from kernel...done.

(gdb) break swtch

Breakpoint 1 at 0x8010479b: file swtch.S, line 11.

(gdb) continue

Continuing.

Thread 1 hit Breakpoint 1, swtch () at swtch.S:11

11 movl 4(%esp), %eax

(gdb) step

12 movl 8(%esp), %edx

(gdb) step

15 pushl %ebp

(gdb) step

swtch () at swtch.S:16

16 pushl %ebx

(gdb) step

swtch () at swtch.S:17

17 pushl %esi

(gdb) step

swtch () at swtch.S:18

18 pushl %edi

(gdb) step

swtch () at swtch.S:21

21 movl %esp, (%eax)

(gdb) step

22 movl %edx, %esp

(gdb) step

swtch () at swtch.S:25

25 popl %edi

(gdb) step

swtch () at swtch.S:26

26 popl %esi

(gdb) step

swtch () at swtch.S:27

27 popl %ebx

(gdb) step

swtch () at swtch.S:28

28 popl %ebp

(gdb) step

swtch () at swtch.S:29

29 ret

(gdb) step

forkret () at proc.c:401

401 release(&ptable.lock);

(gdb) step

release (lk=0x80112d20 <ptable>) at spinlock.c:49

49 if(!holding(lk))

(gdb) step

holding (lock=0x80112d20 <ptable>) at spinlock.c:94

94 r = lock->locked && lock->cpu == mycpu();

(gdb) step

mycpu () at x86.h:99

99 return eflags;

(gdb) step

45 apicid = lapicid();

(gdb) step

lapicid () at lapic.c:103

103 if (!lapic)

(gdb) step

105 return lapic[ID] >> 24;

(gdb) continue

Continuing.

Thread 1 hit Breakpoint 1, swtch () at swtch.S:11

11 movl 4(%esp), %eax

(gdb) clear

Deleted breakpoint 1

(gdb) break exec

Breakpoint 2 at 0x80100a80: file exec.c, line 20.

(gdb) continue

Continuing.

[Switching to Thread 2]

Thread 2 hit Breakpoint 2, exec (path=0x1c "/init", argv=0x8dfffed0)

at exec.c:20

20 struct proc \*curproc = myproc();

(gdb) continue

Continuing.

Thread 2 hit Breakpoint 2, exec (path=0x846 "sh", argv=0x8dffeed0) at exec.c:20

20 struct proc \*curproc = myproc();

(gdb) continue

Continuing.

[Switching to Thread 1]

Thread 1 hit Breakpoint 2, exec (path=0x18e0 "ssls", argv=0x8dfbeed0)

at exec.c:20

20 struct proc \*curproc = myproc();

(gdb) quit

A debugging session is active.

Inferior 1 [Remote target] will be detached.

Quit anyway? (y or n) y

Detaching from program: /home/yjarrar/Documents/Lab 1/xv6-public/kernel, Remote target

Ending remote debugging.

[Inferior 1 (Remote target) detached]

[yjarrar@linux xv6-public]$ make

make: 'xv6.img' is up to date.

[yjarrar@linux xv6-public]$ make qemu-nox

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o cp.o cp.c

cp.c: In function ‘main’:

cp.c:10:12: error: comparison between pointer and integer [-Werror]

if (argv <= 2) {

^~

cc1: all warnings being treated as errors

make: \*\*\* [<builtin>: cp.o] Error 1

[yjarrar@linux xv6-public]$ make

make: 'xv6.img' is up to date.

[yjarrar@linux xv6-public]$ make qemu-nox

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o cp.o cp.c

ld -m elf\_i386 -N -e main -Ttext 0 -o \_cp cp.o ulib.o usys.o printf.o umalloc.o

objdump -S \_cp > cp.asm

objdump -t \_cp | sed '1,/SYMBOL TABLE/d; s/ .\* / /; /^$/d' > cp.sym

./mkfs fs.img README \_cat \_echo \_forktest \_grep \_init \_kill \_ln \_ls \_mkdir \_rm \_sh \_stressfs \_usertests \_wc \_cp \_zombie

nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 941 total 1000

balloc: first 693 blocks have been allocated

balloc: write bitmap block at sector 58

which: no qemu in (/usr/share/Modules/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/yjarrar/.local/bin:/home/yjarrar/bin)

qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512

SeaBIOS (version ?-20180724\_192412-buildhw-07.phx2.fedoraproject.org-1.fc29)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF913E0+1FED13E0 C980

Booting from Hard Disk..xv6...

cpu1: starting 1

cpu0: starting 0

sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8

init: starting sh

$ ls

. 1 1 512

.. 1 1 512

README 2 2 2327

cat 2 3 16164

echo 2 4 14980

forktest 2 5 9260

grep 2 6 18432

init 2 7 15600

kill 2 8 15016

ln 2 9 14900

ls 2 10 17504

mkdir 2 11 15132

rm 2 12 15108

sh 2 13 27668

stressfs 2 14 15952

usertests 2 15 65948

wc 2 16 16856

cp 2 17 15824

zombie 2 18 14716

console 3 19 0

$ cp

Need 2 Arguements!

$ cp README myFile

$ ls

. 1 1 512

.. 1 1 512

README 2 2 2327

cat 2 3 16164

echo 2 4 14980

forktest 2 5 9260

grep 2 6 18432

init 2 7 15600

kill 2 8 15016

ln 2 9 14900

ls 2 10 17504

mkdir 2 11 15132

rm 2 12 15108

sh 2 13 27668

stressfs 2 14 15952

usertests 2 15 65948

wc 2 16 16856

cp 2 17 15824

zombie 2 18 14716

console 3 19 0

myFile 2 20 2327

$ cat myFile

exec: fail

exec File failed

$ cat myFile

xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix

Version 6 (v6). xv6 loosely follows the structure and style of v6,

but is implemented for a modern x86-based multiprocessor using ANSI C.

ACKNOWLEDGMENTS

xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer

to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,

2000)). See also https://pdos.csail.mit.edu/6.828/, which

provides pointers to on-line resources for v6.

xv6 borrows code from the following sources:

JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)

Plan 9 (entryother.S, mp.h, mp.c, lapic.c)

FreeBSD (ioapic.c)

NetBSD (console.c)

The following people have made contributions: Russ Cox (context switching,

locking), Cliff Frey (MP), Xiao Yu (MP), Nickolai Zeldovich, and Austin

Clements.

We are also grateful for the bug reports and patches contributed by Silas

Boyd-Wickizer, Anton Burtsev, Cody Cutler, Mike CAT, Tej Chajed, eyalz800,

Nelson Elhage, Saar Ettinger, Alice Ferrazzi, Nathaniel Filardo, Peter

Froehlich, Yakir Goaron,Shivam Handa, Bryan Henry, Jim Huang, Alexander

Kapshuk, Anders Kaseorg, kehao95, Wolfgang Keller, Eddie Kohler, Austin

Liew, Imbar Marinescu, Yandong Mao, Matan Shabtay, Hitoshi Mitake, Carmi

Merimovich, Mark Morrissey, mtasm, Joel Nider, Greg Price, Ayan Shafqat,

Eldar Sehayek, Yongming Shen, Cam Tenny, tyfkda, Rafael Ubal, Warren

Toomey, Stephen Tu, Pablo Ventura, Xi Wang, Keiichi Watanabe, Nicolas

Wolovick, wxdao, Grant Wu, Jindong Zhang, Icenowy Zheng, and Zou Chang Wei.

The code in the files that constitute xv6 is

Copyright 2006-2018 Frans Kaashoek, Robert Morris, and Russ Cox.

ERROR REPORTS

Please send errors and suggestions to Frans Kaashoek and Robert Morris

(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching

operating system for MIT's 6.828, so we are more interested in

simplifications and clarifications than new features.

BUILDING AND RUNNING XV6

To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run

"make". On non-x86 or non-ELF machines (like OS X, even on x86), you

will need to install a cross-compiler gcc suite capable of producing

x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).

Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC

simulator and run "make qemu".

$ cp README myFile1 myFile2

$ cat myFile1

xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix

Version 6 (v6). xv6 loosely follows the structure and style of v6,

but is implemented for a modern x86-based multiprocessor using ANSI C.

ACKNOWLEDGMENTS

xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer

to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,

2000)). See also https://pdos.csail.mit.edu/6.828/, which

provides pointers to on-line resources for v6.

xv6 borrows code from the following sources:

JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)

Plan 9 (entryother.S, mp.h, mp.c, lapic.c)

FreeBSD (ioapic.c)

NetBSD (console.c)

The following people have made contributions: Russ Cox (context switching,

locking), Cliff Frey (MP), Xiao Yu (MP), Nickolai Zeldovich, and Austin

Clements.

We are also grateful for the bug reports and patches contributed by Silas

Boyd-Wickizer, Anton Burtsev, Cody Cutler, Mike CAT, Tej Chajed, eyalz800,

Nelson Elhage, Saar Ettinger, Alice Ferrazzi, Nathaniel Filardo, Peter

Froehlich, Yakir Goaron,Shivam Handa, Bryan Henry, Jim Huang, Alexander

Kapshuk, Anders Kaseorg, kehao95, Wolfgang Keller, Eddie Kohler, Austin

Liew, Imbar Marinescu, Yandong Mao, Matan Shabtay, Hitoshi Mitake, Carmi

Merimovich, Mark Morrissey, mtasm, Joel Nider, Greg Price, Ayan Shafqat,

Eldar Sehayek, Yongming Shen, Cam Tenny, tyfkda, Rafael Ubal, Warren

Toomey, Stephen Tu, Pablo Ventura, Xi Wang, Keiichi Watanabe, Nicolas

Wolovick, wxdao, Grant Wu, Jindong Zhang, Icenowy Zheng, and Zou Chang Wei.

The code in the files that constitute xv6 is

Copyright 2006-2018 Frans Kaashoek, Robert Morris, and Russ Cox.

ERROR REPORTS

Please send errors and suggestions to Frans Kaashoek and Robert Morris

(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching

operating system for MIT's 6.828, so we are more interested in

simplifications and clarifications than new features.

BUILDING AND RUNNING XV6

To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run

"make". On non-x86 or non-ELF machines (like OS X, even on x86), you

will need to install a cross-compiler gcc suite capable of producing

x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).

Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC

simulator and run "make qemu".

$ cat myFile2

cat: cannot open myFile2

$ cat myFile2

cat: cannot open myFile2

$ vi myFile2

exec: fail

exec vi failed

$ cp README myFile2

$ cat myFile2

xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix

Version 6 (v6). xv6 loosely follows the structure and style of v6,

but is implemented for a modern x86-based multiprocessor using ANSI C.

ACKNOWLEDGMENTS

xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer

to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,

2000)). See also https://pdos.csail.mit.edu/6.828/, which

provides pointers to on-line resources for v6.

xv6 borrows code from the following sources:

JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)

Plan 9 (entryother.S, mp.h, mp.c, lapic.c)

FreeBSD (ioapic.c)

NetBSD (console.c)

The following people have made contributions: Russ Cox (context switching,

locking), Cliff Frey (MP), Xiao Yu (MP), Nickolai Zeldovich, and Austin

Clements.

We are also grateful for the bug reports and patches contributed by Silas

Boyd-Wickizer, Anton Burtsev, Cody Cutler, Mike CAT, Tej Chajed, eyalz800,

Nelson Elhage, Saar Ettinger, Alice Ferrazzi, Nathaniel Filardo, Peter

Froehlich, Yakir Goaron,Shivam Handa, Bryan Henry, Jim Huang, Alexander

Kapshuk, Anders Kaseorg, kehao95, Wolfgang Keller, Eddie Kohler, Austin

Liew, Imbar Marinescu, Yandong Mao, Matan Shabtay, Hitoshi Mitake, Carmi

Merimovich, Mark Morrissey, mtasm, Joel Nider, Greg Price, Ayan Shafqat,

Eldar Sehayek, Yongming Shen, Cam Tenny, tyfkda, Rafael Ubal, Warren

Toomey, Stephen Tu, Pablo Ventura, Xi Wang, Keiichi Watanabe, Nicolas

Wolovick, wxdao, Grant Wu, Jindong Zhang, Icenowy Zheng, and Zou Chang Wei.

The code in the files that constitute xv6 is

Copyright 2006-2018 Frans Kaashoek, Robert Morris, and Russ Cox.

ERROR REPORTS

Please send errors and suggestions to Frans Kaashoek and Robert Morris

(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching

operating system for MIT's 6.828, so we are more interested in

simplifications and clarifications than new features.

BUILDING AND RUNNING XV6

To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run

"make". On non-x86 or non-ELF machines (like OS X, even on x86), you

will need to install a cross-compiler gcc suite capable of producing

x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).

Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC

simulator and run "make qemu".

$ clear

exec: fail

exec clear failed

$

$

$

$

$

$

$ ls

. 1 1 512

.. 1 1 512

README 2 2 2327

cat 2 3 16164

echo 2 4 14980

forktest 2 5 9260

grep 2 6 18432

init 2 7 15600

kill 2 8 15016

ln 2 9 14900

ls 2 10 17504

mkdir 2 11 15132

rm 2 12 15108

sh 2 13 27668

stressfs 2 14 15952

usertests 2 15 65948

wc 2 16 16856

cp 2 17 15824

zombie 2 18 14716

console 3 19 0

myFile 2 20 2327

myFile1 2 21 2327

myFile2 2 22 2327

$