Advanced Programming in the UNIX Environment

Week 06, Segment 2: Program Startup

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/

ISO/IEC 9899:2018

"5.1.2.2.1 Program startup

The function called at program startup is named **main**. The implementation declares no prototype for this function. It shall be defined with a return type of **int** and with no parameters:

```
int main(void) { /*...*/ }
```

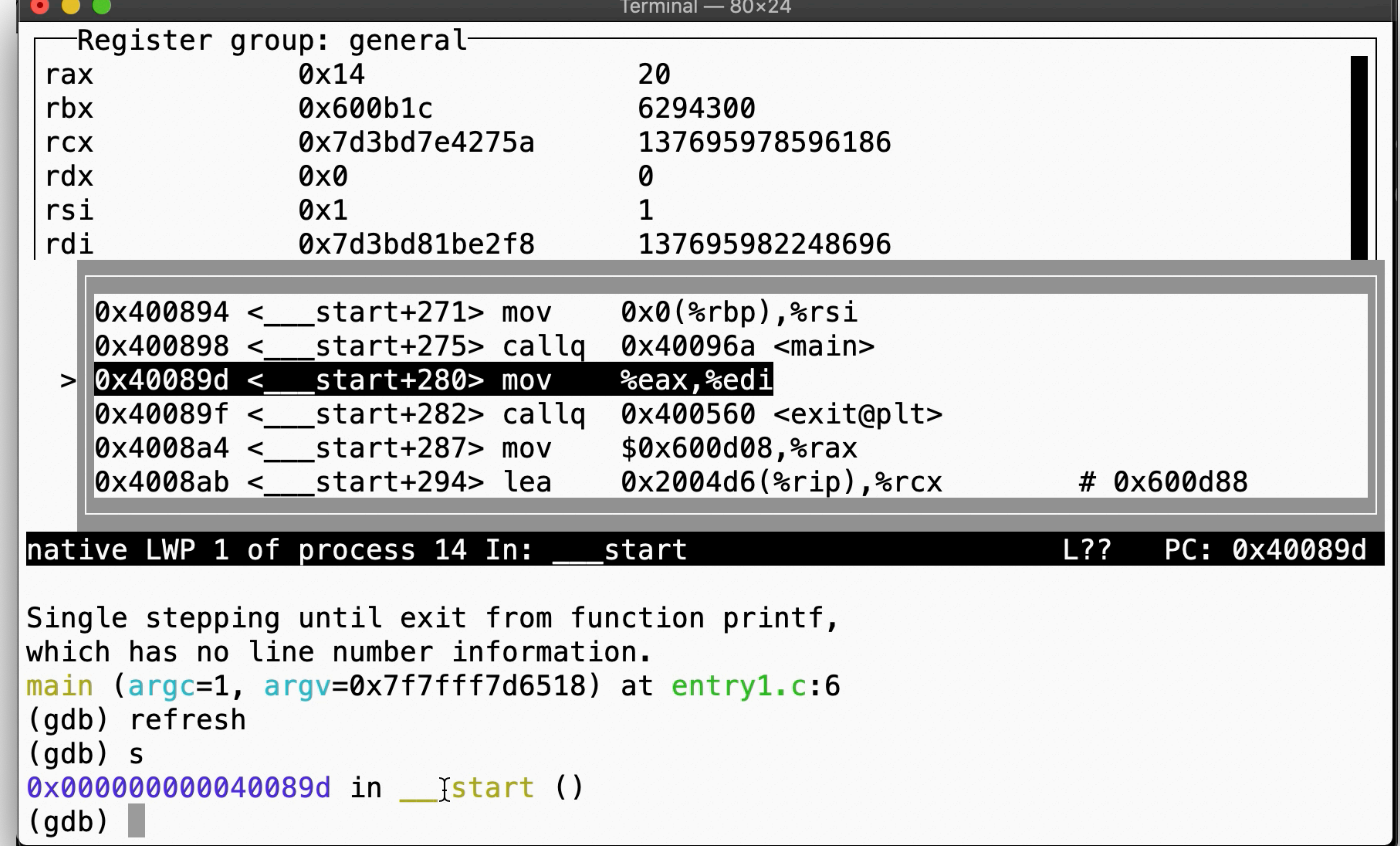
or with two parameters (referred to here as *argc* and *argv*, though any names may be used, as they are local to the function in which they are declared):

```
int main(int argc, char*argv[]) { /*...*/ }
```

or equivalent; *or in some other implementation-defined manner*."

main

- when one of the `exec` functions is called, the kernel needs to start the given program
- special startup routine called by kernel which sets up things for `main` (or whatever entrypoint is defined)
- `argc` is a count of the number of command line arguments (including the command itself)
- `argv` is an array of pointers to the arguments
- it is guaranteed by both ANSI C and POSIX.1 that `argv[argc] == NULL`

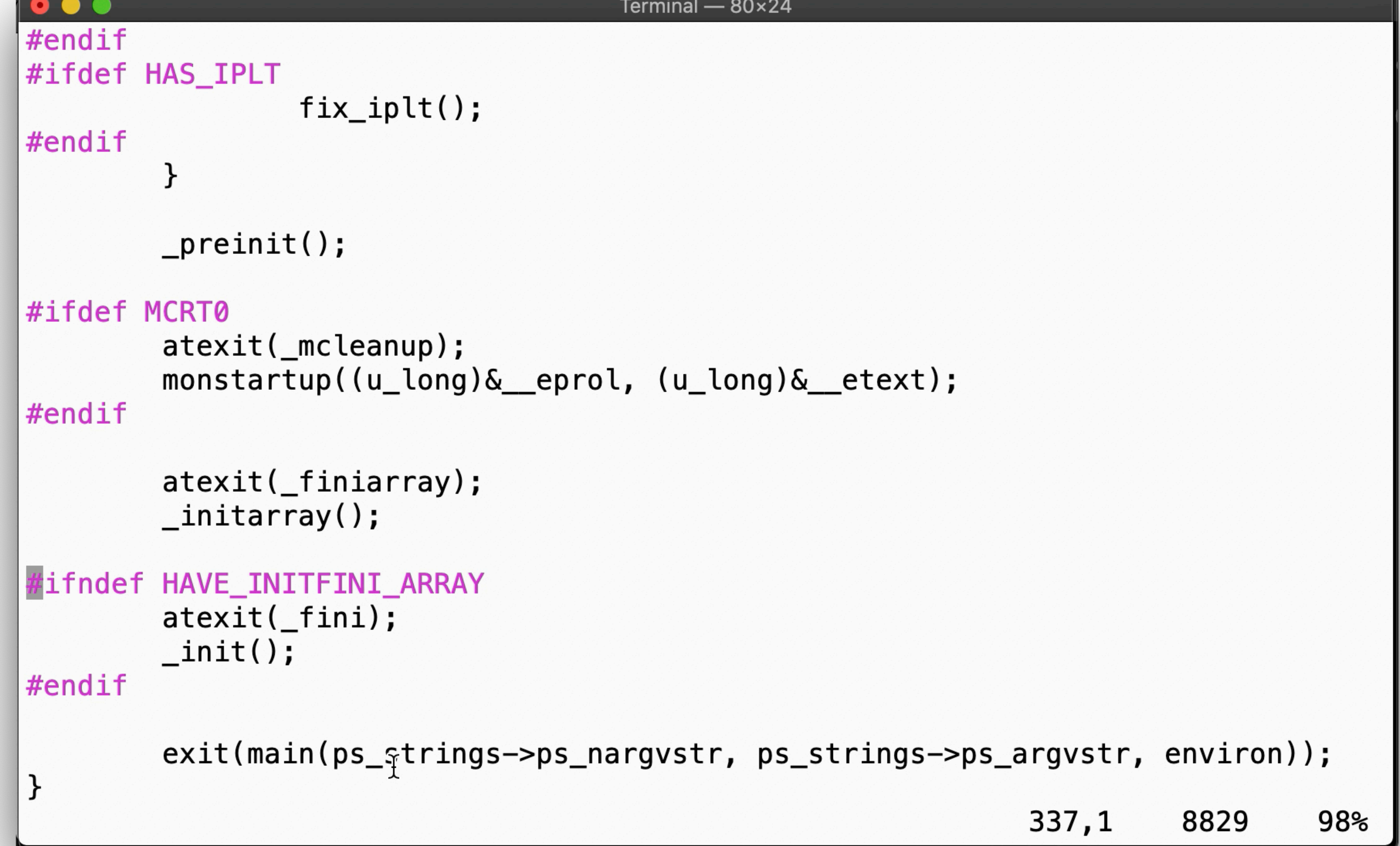


```
Terminal — 80×24
┌Register group: general─────────────────────────────────────────────────────┐
│rax            0x14                 20                                      │
│rbx            0x600b1c             6294300                                 │
│rcx            0x7d3bd7e4275a       137695978596186                         │
│rdx            0x0                  0                                       │
│rsi            0x1                  1                                       │
│rdi            0x7d3bd81be2f8       137695982248696                         │

   0x400894 <___start+271> mov    0x0(%rbp),%rsi
   0x400898 <___start+275> callq  0x40096a <main>
 > 0x40089d <___start+280> mov    %eax,%edi
   0x40089f <___start+282> callq  0x400560 <exit@plt>
   0x4008a4 <___start+287> mov    $0x600d08,%rax
   0x4008ab <___start+294> lea    0x2004d6(%rip),%rcx        # 0x600d88

native LWP 1 of process 14 In: ___start                       L??   PC: 0x40089d

Single stepping until exit from function printf,
which has no line number information.
main (argc=1, argv=0x7f7fff7d6518) at entry1.c:6
(gdb) refresh
(gdb) s
0x000000000040089d in ___start ()
(gdb) 
```



```
#endif
#ifdef HAS_IPLT
                fix_iplt();
#endif
        }

        _preinit();

#ifdef MCRT0
        atexit(_mcleanup);
        monstartup((u_long)&__eprol, (u_long)&__etext);
#endif

        atexit(_finiarray);
        _initarray();

#ifndef HAVE_INITFINI_ARRAY
        atexit(_fini);
        _init();
#endif

        exit(main(ps_strings->ps_nargvstr, ps_strings->ps_argvstr, environ));
}
                                                              337,1     8829     98%
```

ISO/IEC 9899:2018

It shall be defined with a return type of **int** and with no parameters:

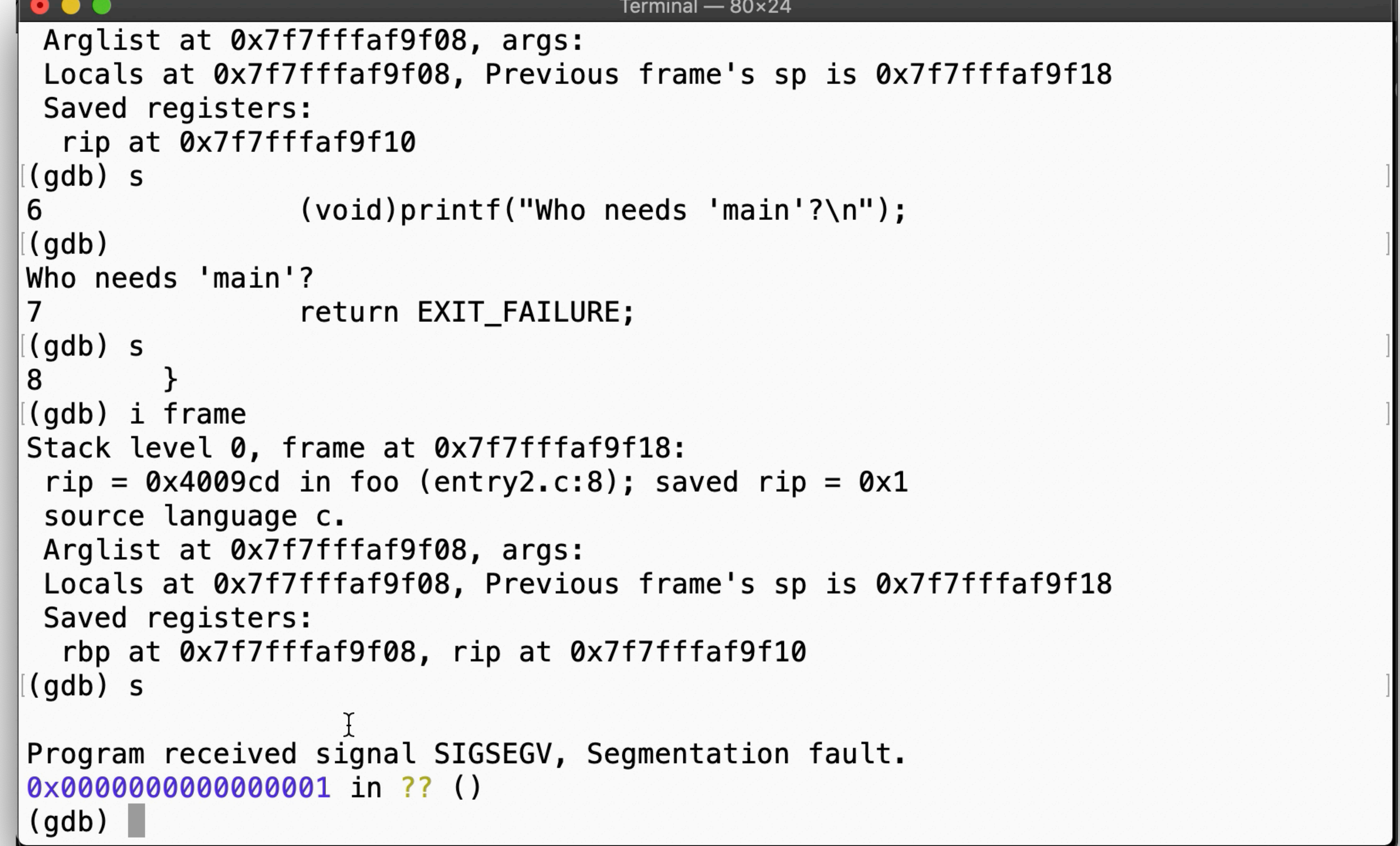
```
int main(void) { /*...*/ }
```

or with two parameters:

```
int main(int argc, char*argv[]) { /*...*/ }
```

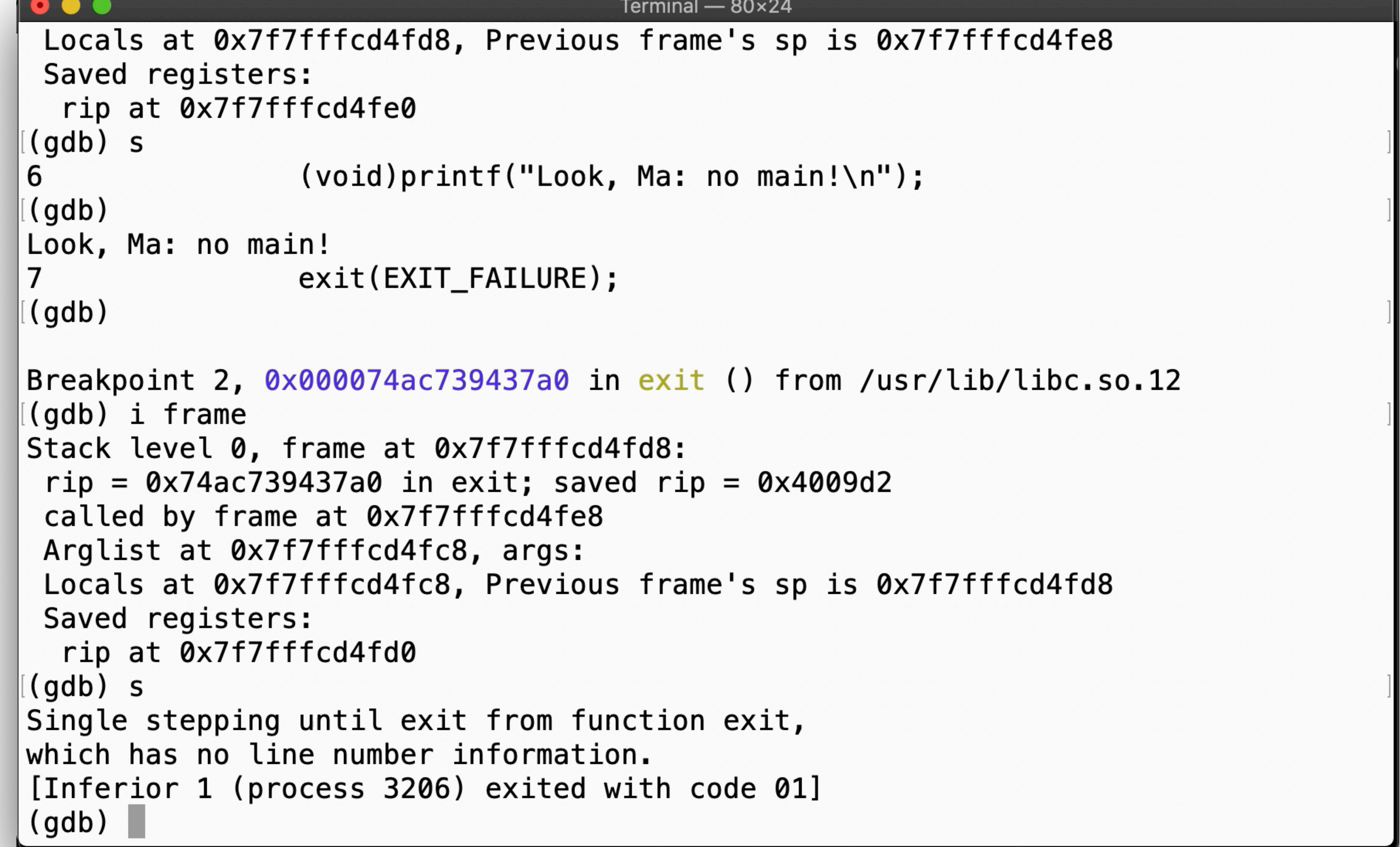
or in some other implementation-defined manner:

```
int main(int argc, char*argv[], char *envp[]) { /*...*/ }
```



```
 Arglist at 0x7f7fffaf9f08, args:
 Locals at 0x7f7fffaf9f08, Previous frame's sp is 0x7f7fffaf9f18
 Saved registers:
  rip at 0x7f7fffaf9f10
(gdb) s
6               (void)printf("Who needs 'main'?\n");
(gdb)
Who needs 'main'?
7               return EXIT_FAILURE;
(gdb) s
8       }
(gdb) i frame
Stack level 0, frame at 0x7f7fffaf9f18:
 rip = 0x4009cd in foo (entry2.c:8); saved rip = 0x1
 source language c.
 Arglist at 0x7f7fffaf9f08, args:
 Locals at 0x7f7fffaf9f08, Previous frame's sp is 0x7f7fffaf9f18
 Saved registers:
  rbp at 0x7f7fffaf9f08, rip at 0x7f7fffaf9f10
(gdb) s


Program received signal SIGSEGV, Segmentation fault.
0x0000000000000001 in ?? ()
(gdb)
```


Terminal — 80×24

```
 Locals at 0x7f7fffcd4fd8, Previous frame's sp is 0x7f7fffcd4fe8
 Saved registers:
  rip at 0x7f7fffcd4fe0
(gdb) s
6               (void)printf("Look, Ma: no main!\n");
(gdb)
Look, Ma: no main!
7               exit(EXIT_FAILURE);
(gdb)

Breakpoint 2, 0x000074ac739437a0 in exit () from /usr/lib/libc.so.12
(gdb) i frame
Stack level 0, frame at 0x7f7fffcd4fd8:
 rip = 0x74ac739437a0 in exit; saved rip = 0x4009d2
 called by frame at 0x7f7fffcd4fe8
 Arglist at 0x7f7fffcd4fc8, args:
 Locals at 0x7f7fffcd4fc8, Previous frame's sp is 0x7f7fffcd4fd8
 Saved registers:
  rip at 0x7f7fffcd4fd0
(gdb) s
Single stepping until exit from function exit,
which has no line number information.
[Inferior 1 (process 3206) exited with code 01]
(gdb) 
```

Program Startup

- The program entry point is defined by the compiler/linker.
- The C startup routine sets up the environment and moves arguments etc. into the right registers for `main` to be called.
- `main` returns an `int`, which is passed to `exit(3)`

When a program is started, we don't just call main(); instead we observed

```
_start() -> ___start() -> exit(main(...))
```

Links

- https://stackoverflow.com/questions/7976433/debugging-the-c-runtime
- https://embeddedartistry.com/blog/2019/04/08/a-general-overview-of-what-happens-before-main/
- http://articles.manugarg.com/aboutelfauxiliaryvectors
- https://blogs.oracle.com/linux/hello-from-a-libc-free-world-part-1-v2
- https://www.recurse.com/blog/7-understanding-c-by-learning-assembly
- https://manybutfinite.com/post/journey-to-the-stack/

- https://stevens.netmeister.org/631/startup-exercise.html