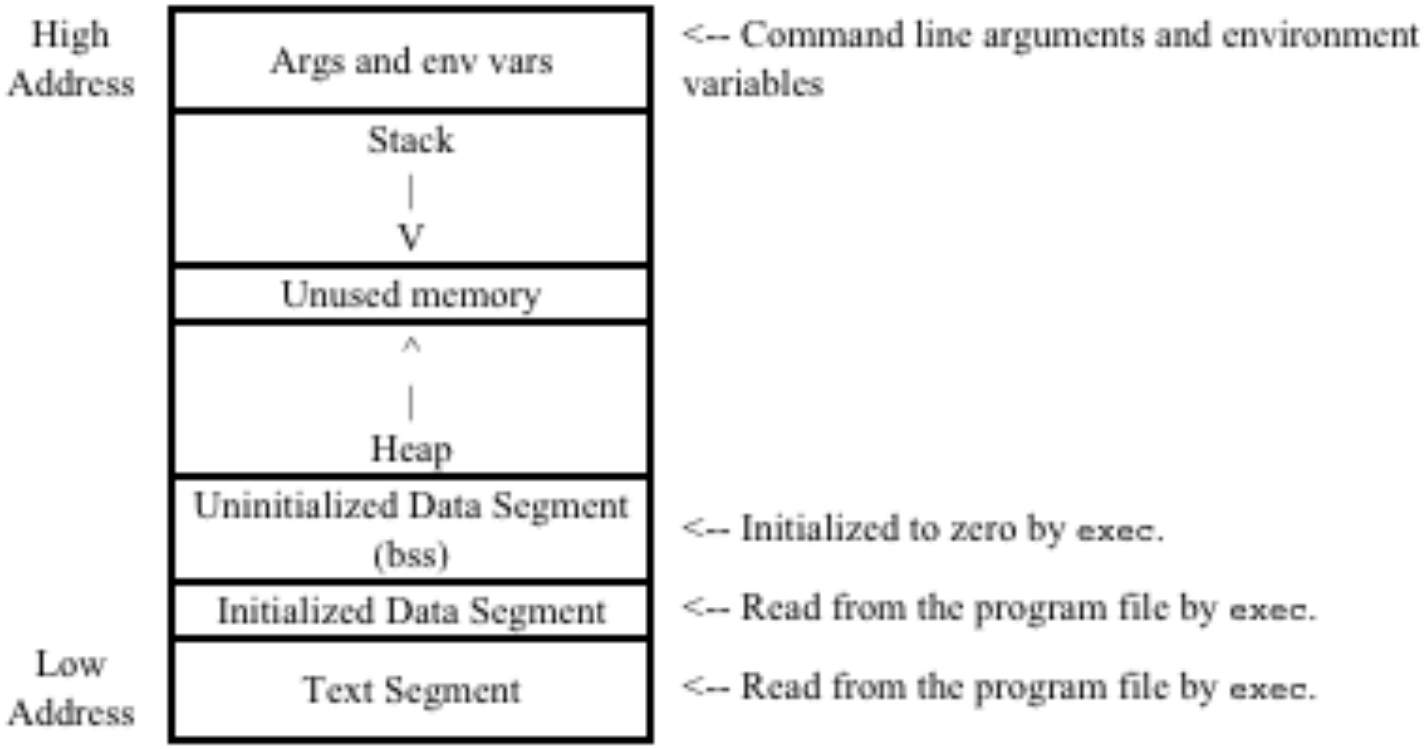
Debugging and Memory Layout

ASTR615 – Ricotti

Memory Layout



High Address

Args and env vars

<-- Command line arguments and environment variables

Stack
|
V

Unused memory

^
|
Heap

Uninitialized Data Segment (bss)

<-- Initialized to zero by exec.

Initialized Data Segment

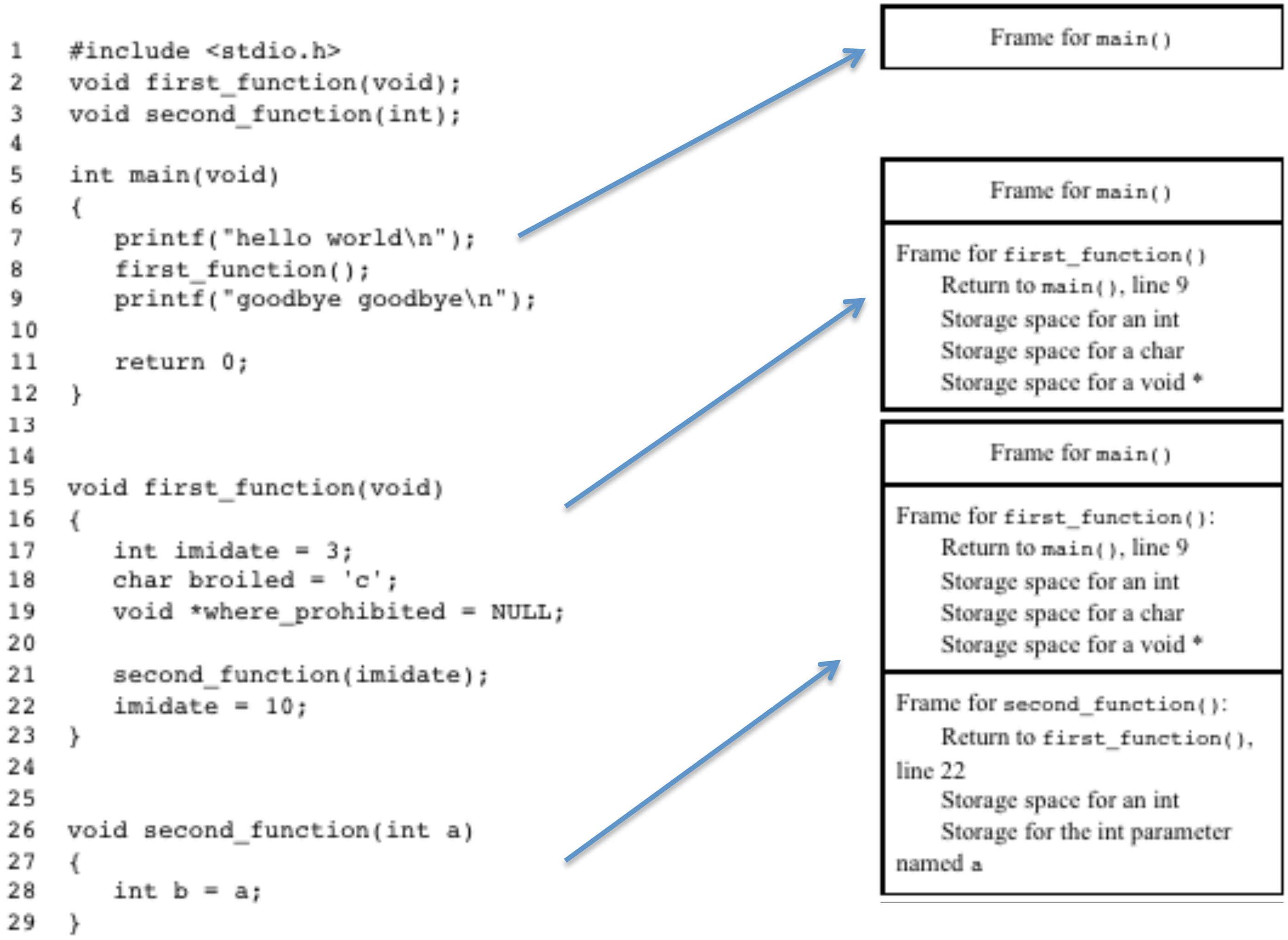
<-- Read from the program file by exec.

Low Address

Text Segment

<-- Read from the program file by exec.

Stack size and frames



```
1   #include <stdio.h>
2   void first_function(void);
3   void second_function(int);
4
5   int main(void)
6   {
7      printf("hello world\n");
8      first_function();
9      printf("goodbye goodbye\n");
10
11     return 0;
12  }
13
14
15  void first_function(void)
16  {
17     int imidate = 3;
18     char broiled = 'c';
19     void *where_prohibited = NULL;
20
21     second_function(imidate);
22     imidate = 10;
23  }
24
25
26  void second_function(int a)
27  {
28     int b = a;
29  }
```

Frame for main()

Frame for main()

Frame for first_function()
Return to main(), line 9
Storage space for an int
Storage space for a char
Storage space for a void *

Frame for main()

Frame for first_function():
Return to main(), line 9
Storage space for an int
Storage space for a char
Storage space for a void *

Frame for second_function():
Return to first_function(), line 22
Storage space for an int
Storage for the int parameter named a

Exercises

Check the stack size on your computer with

- limit

Check the size of the executable with

- size ./file

text, data (initialized vars), bss (uninizialized vars)

Step 1

1. cc -o debug debug.c -g -lm
2. gdb ./debug

-g or –ggdb loads the *enhanced symbol table (*necessary for debugging)

PS: -g option can also be used with optimizations (not recommended) and will not slow down the code

Step 2: basic commands

- r (run code)
- q (quit)
- breakpoint # (breakpoint at line #)
- bt (print stack)
- l (list code)
- frame #
- step -> step line of code
- next -> same (skip content of functions)
- print i
- set var i=xx

Attach GDB to a running job

```
•    gdb beer-process 2764

•    bt (backtrace)
•    (gdb) bt
•      #0  0x410c64fb in nanosleep () from /lib/tls/libc.so.6
•      #1  0x410c6358 in sleep () from /lib/tls/libc.so.6
•      #2  0x0804841f in GoToSleep () at beer-process.c:32
•      #3  0x080483e0 in main () at beer-process.c:14
•    (gdb) frame 3
•      #3  0x080483eb in main () at beer-process.c:15
•      15                    GoToSleep();
•      (gdb) print i
•      $1 = 99997
•    (gdb) next
•      Single stepping until exit from function nanosleep,
•      which has no line number information.
•      0x410c6358 in sleep () from /lib/tls/libc.so.6
•      (gdb) step
•      Single stepping until exit from function sleep,
•      which has no line number information.
•      GoToSleep () at beer-process.c:34
•    34      }
•      (gdb) bt
•      #0  GoToSleep () at beer-process.c:34
•      #1  0x080483eb in main () at beer-process.c:15

•    HERE IS THE COOL PART
•     (gdb) frame 3
•      #3  0x080483eb in main () at beer-process.c:15
•      15                    GoToSleep();
•      (gdb) set var i = 99999999

•    quit!
```