Computer Organization
Using gdb with tui

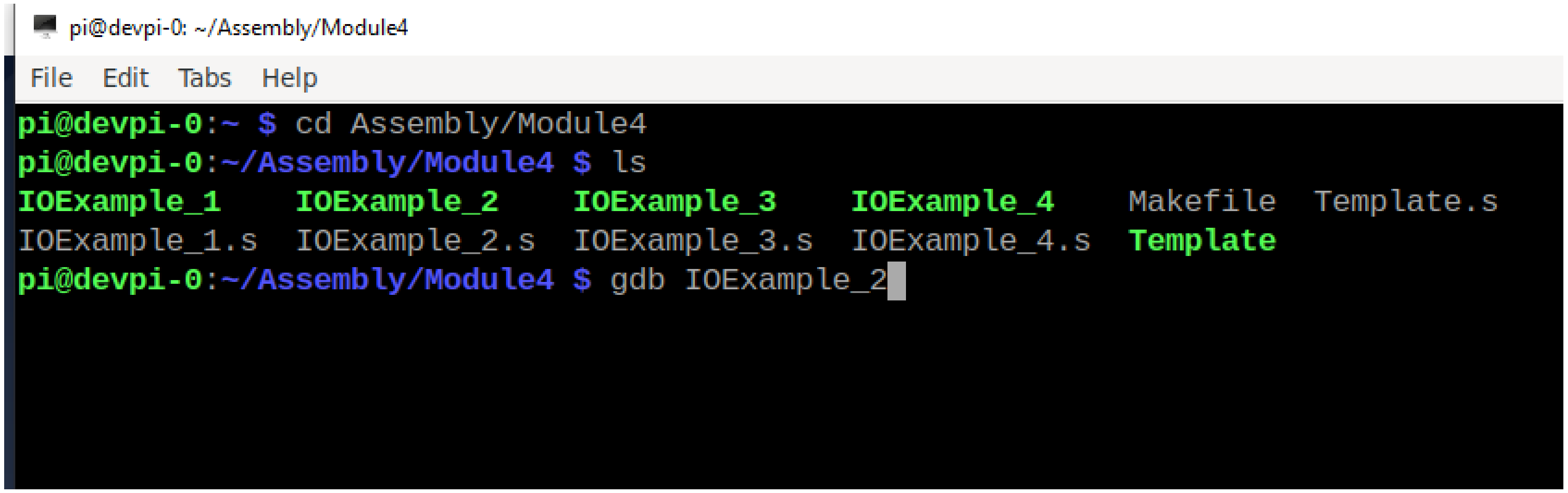
Prof. Charles W. Kann

Overview of Class

- Why use C libraries for I/O
- Writing HelloWorld
- Prompt for, retrieve,and print a string
- Prompt for, retrieve, and print an int
  - With a static variable
  - Using a stack variable
- Makefile and touch

Starting gdb

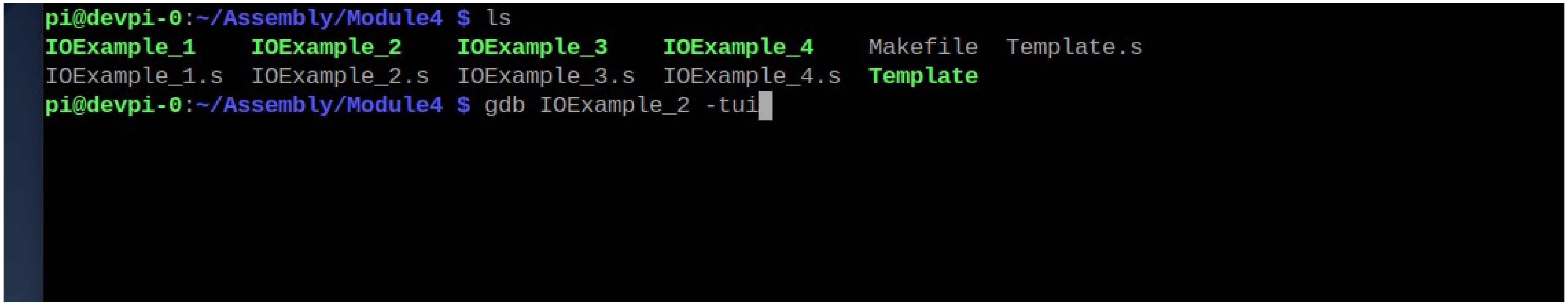
- Make your screen large (wide and long)
- Run “gdb *executable* -tui”
- For this example, use “gdb IOExample_2 -tui)



```
pi@devpi-0:~ $ cd Assembly/Module4
pi@devpi-0:~/Assembly/Module4 $ ls
IOExample_1    IOExample_2    IOExample_3    IOExample_4    Makefile  Template.s
IOExample_1.s  IOExample_2.s  IOExample_3.s  IOExample_4.s  Template
pi@devpi-0:~/Assembly/Module4 $ gdb IOExample_2
```

Starting gdb

- Make your screen large (wide and long)
- Run "gdb *executable* -tui"



```
pi@devpi-0:~/Assembly/Module4 $ ls
IOExample_1     IOExample_2     IOExample_3     IOExample_4     Makefile  Template.s
IOExample_1.s   IOExample_2.s   IOExample_3.s   IOExample_4.s   Template
pi@devpi-0:~/Assembly/Module4 $ gdb IOExample_2 -tui
```

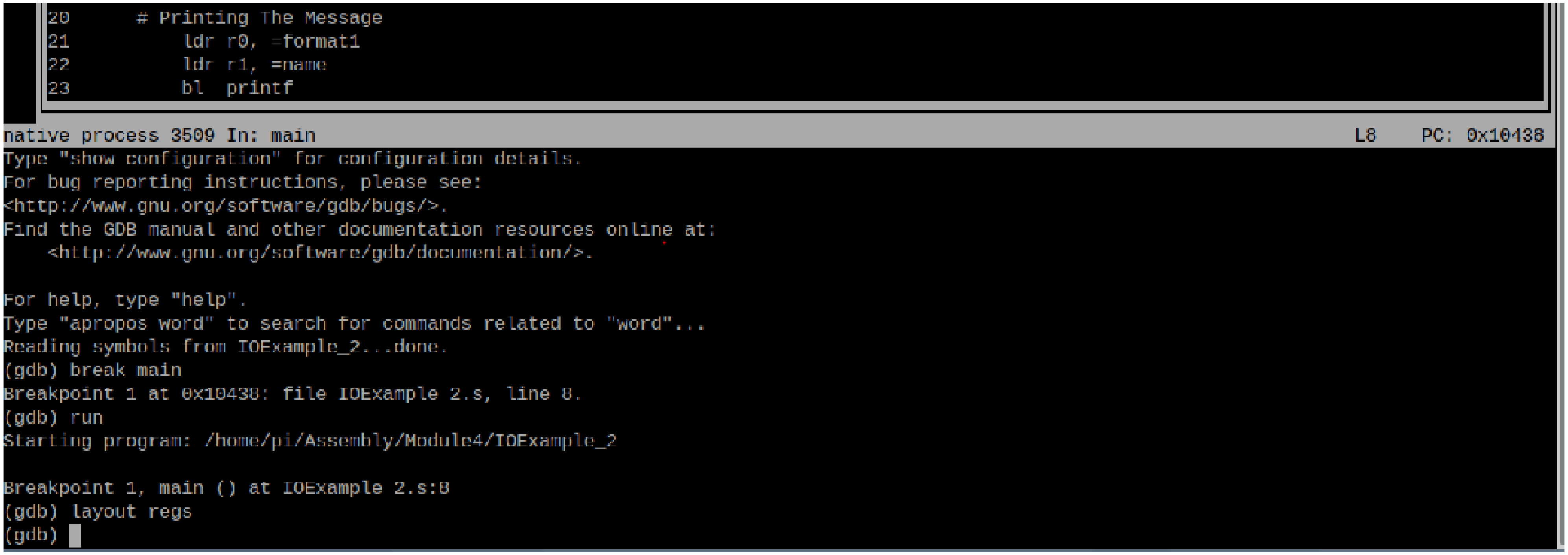
Starting gdb

- IN the console window run the following commands

  break main

  run

  Layout regs

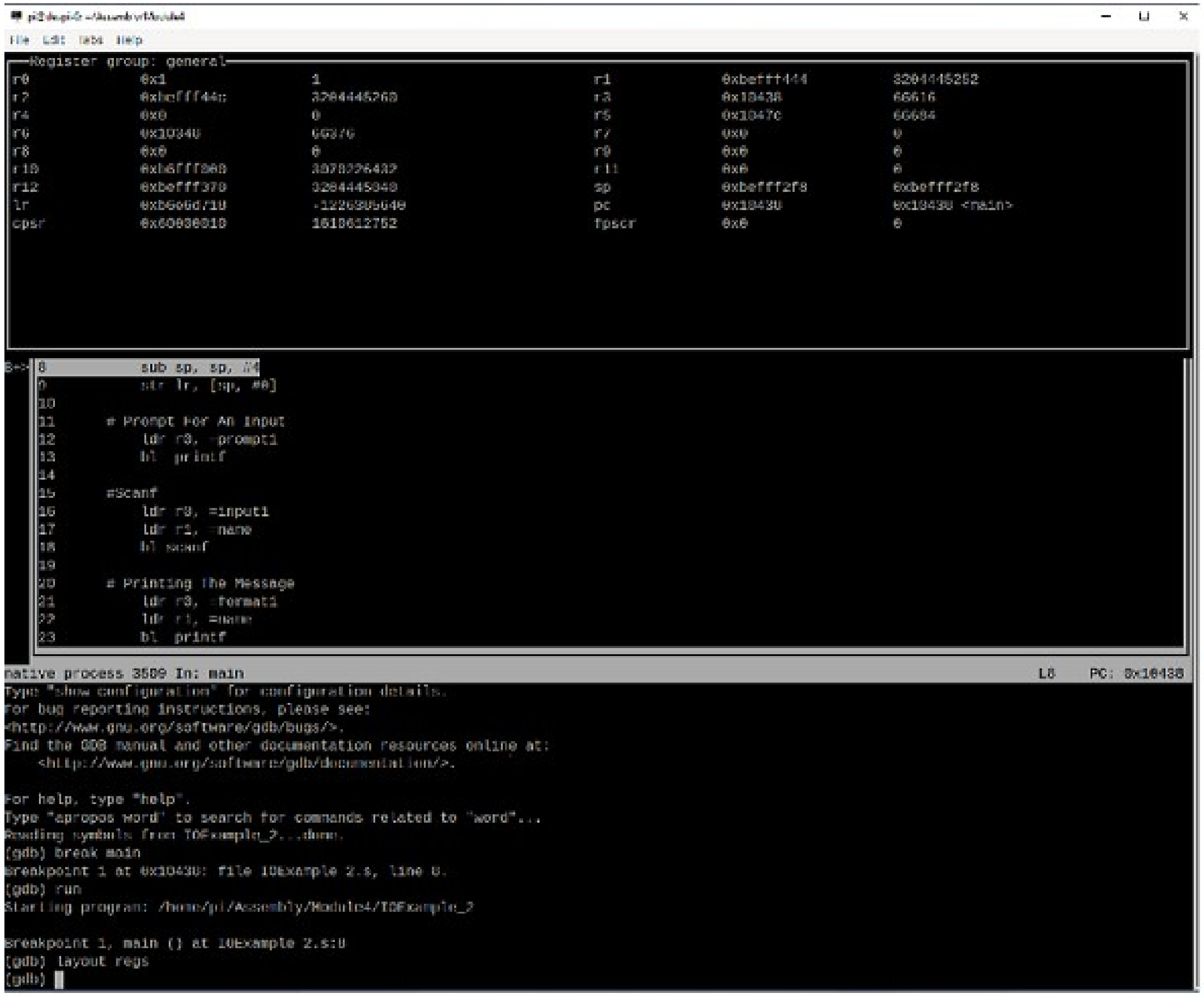


```
20        # Printing The Message
21            ldr r0, =format1
22            ldr r1, =name
23            bl  printf

native process 3509 In: main                                                                     L8    PC: 0x10438
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 IOExample_2...done.
(gdb) break main
Breakpoint 1 at 0x10438: file IOExample 2.s, line 8.
(gdb) run
Starting program: /home/pi/Assembly/Module4/IOExample_2

Breakpoint 1, main () at IOExample 2.s:8
(gdb) layout regs
(gdb) 
```

You should see

- **A screen with three sections**
  - Registers
  - Source code with breakpoints and current position in code
  - Console window
- **See next slide**

Screen image



```
—Register group: general—
r0             0x1                 1                              r1             0xbefff444          3204445252
r2             0xbefff44c          3204445260                     r3             0x10438             66616
r4             0x0                 0                              r5             0x1047c             66684
r6             0x10340             66376                          r7             0x0                 0
r8             0x0                 0                              r9             0x0                 0
r10            0xb6fff000          3078226432                     r11            0x0                 0
r12            0xbefff370          3204445040                     sp             0xbefff2f8          0xbefff2f8
lr             0xb6e6d718          -1226385640                    pc             0x10438             0x10438 <main>
cpsr           0x60000010          1610612752                     fpscr          0x0                 0

B+>8            sub sp, sp, #4
   9            str lr, [sp, #0]
   10
   11       # Prompt For An Input
   12           ldr r0, =prompt1
   13           bl  printf
   14
   15       #Scanf
   16           ldr r0, =input1
   17           ldr r1, =name
   18           bl scanf
   19
   20       # Printing The Message
   21           ldr r0, =format1
   22           ldr r1, =name
   23           bl  printf

native process 3589 In: main                                                                      L8    PC: 0x10438
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 IOExample_2...done.
(gdb) break main
Breakpoint 1 at 0x10438: file IOExample_2.s, line 8.
(gdb) run
Starting program: /home/pi/Assembly/Module4/IOExample_2

Breakpoint 1, main () at IOExample_2.s:8
(gdb) layout regs
(gdb) 
```

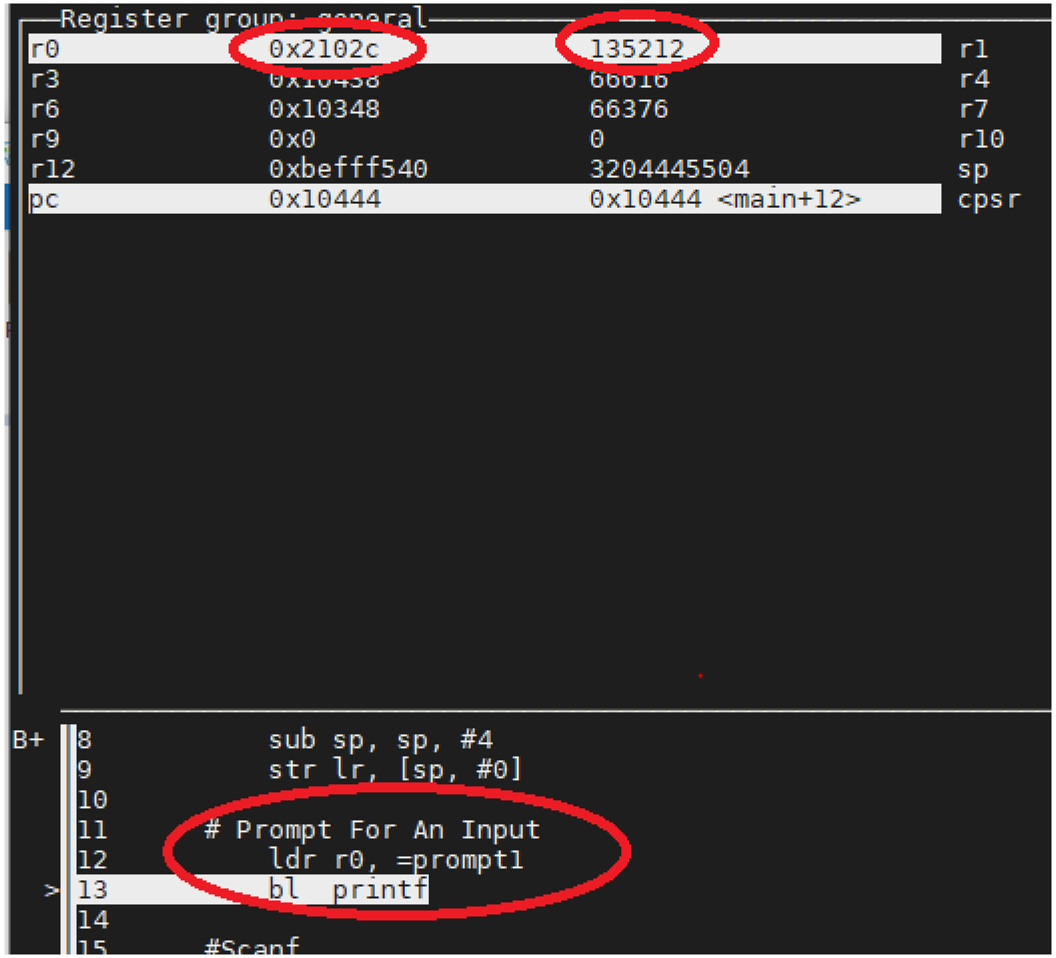
If you need help

- In the command window type “help”
- Note that this is a production debugger (e.g. a real debugger used by real programmers, for real work). There are a lot of options.
- You are welcome to play around with commands, etc., but this module only intends you to use and understand the commands it presents.

To walk through the code

- Type “skip printf” and “skip scanf”. You do not have source code, so this means do not walk through statements in those functions.
- Type “next”. This will move the cursor one step through your program
- Now each time you hit the <Enter> key, you will run the last command, which was next. So you will walk through your program.
- Note that the registers will change when you reach lines in the code that changes them.
- For example, stop the program after “ldr r0” changes the value of the r0 to be the address of prompt1.

r0 after the call to “ldr r0, =prompt1”



```
Register group: general
r0             0x2102c             135212                 r1
r3             0x10438             66616                  r4
r6             0x10348             66376                  r7
r9             0x0                 0                      r10
r12            0xbefff540          3204445504             sp
pc             0x10444             0x10444 <main+12>      cpsr

B+            sub sp, sp, #4
              str lr, [sp, #0]

          # Prompt For An Input
              ldr r0, =prompt1
  >           bl  printf

          #Scanf
```

Printing out the address and value of prompt1

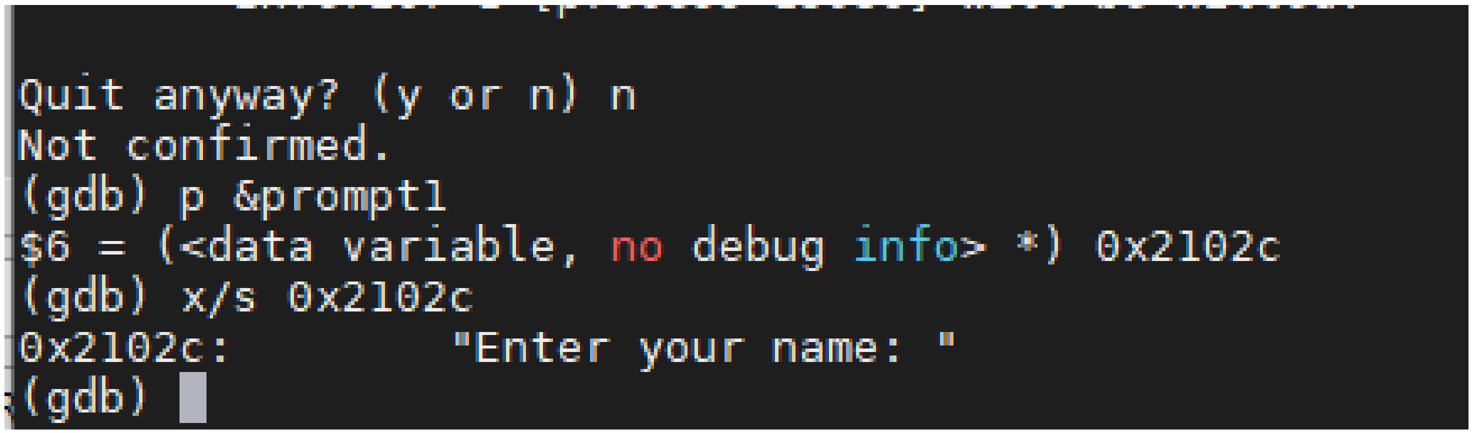
- To find the address of the variable, use the "&" sign. For example, the address or prompt1 can be found by saying:

  print &prompt1

- You can just type "p" instead of print.
- Looking at the last slide and what is printed out here, they agree that the address is 0x2102c
- To print the string, use the x command. x/s prints the string at the address specified, so use "x/s 0x2102c"
- Other format characters can be found on the cheat sheet at:

  https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf

Print and x commands



```
Quit anyway? (y or n) n
Not confirmed.
(gdb) p &prompt1
$6 = (<data variable, no debug info> *) 0x2102c
(gdb) x/s 0x2102c
0x2102c:        "Enter your name: "
(gdb) 
```

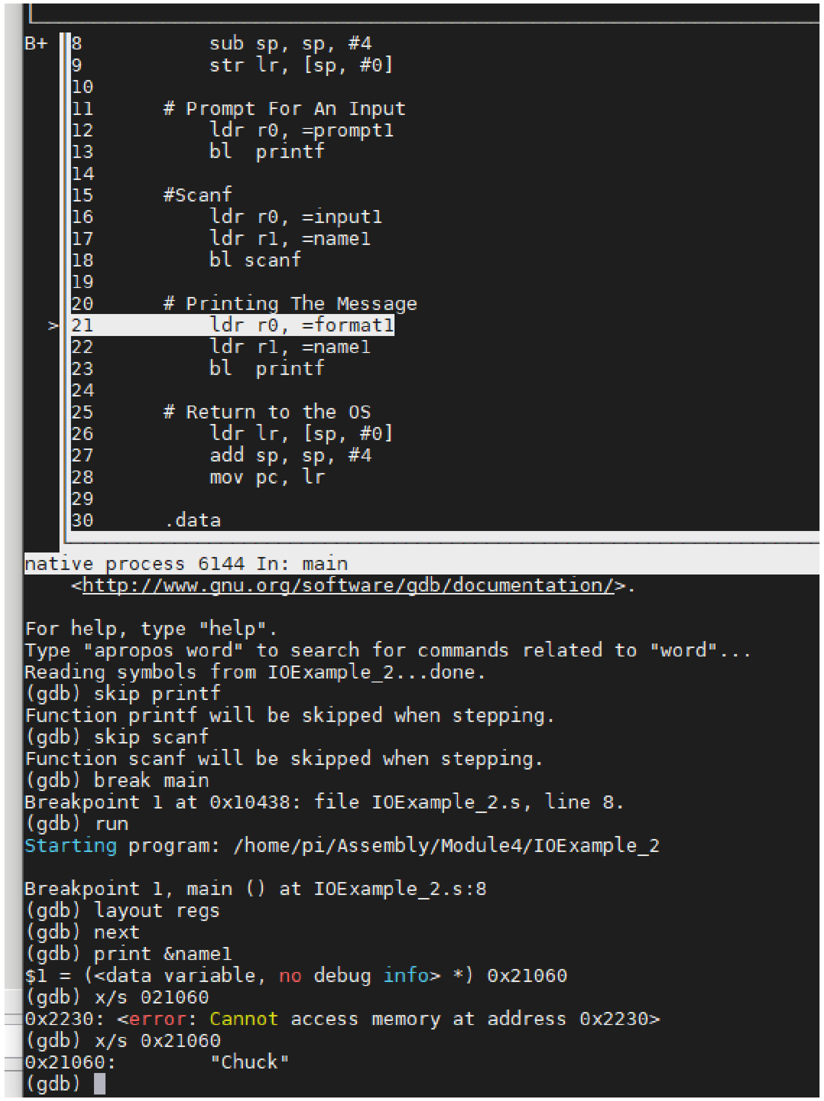
If the UI gets messed up

- Type “skip printf” and “skip scanf”. This will skip these functions when they are called. There is no source code anyway, so you should always skip them.
- Type “next” in the command window. This will execute the instruction “bl printf”, but not stop in printf.
- Note that the “bl printf” instruction causes the screen to get messed up.
- When input or output is taken from the console screen, the program listing portion of the screen will get messed up.
- Typing <ctrl>l will clean up the screen.

Continuing GDB

- Typing <enter> just keeps doing the last action (which was next). Type <enter> until you return from scanf (you will be on the line after scanf).
- Type <ctrl-l> to restore the screen.
- Type print &name to get the address of the name
- Type “x/s ”, which should show the string you entered.

Displaying your input



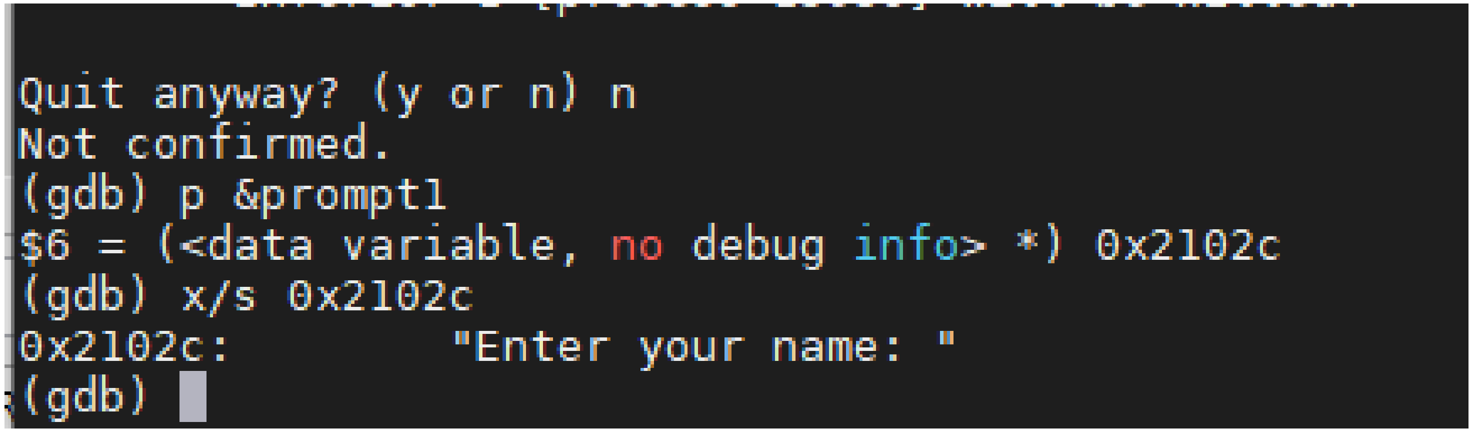
```
B+      sub sp, sp, #4
        str lr, [sp, #0]

    # Prompt For An Input
        ldr r0, =prompt1
        bl  printf

    #Scanf
        ldr r0, =input1
        ldr r1, =name1
        bl scanf

    # Printing The Message
  >     ldr r0, =format1
        ldr r1, =name1
        bl  printf

    # Return to the OS
        ldr lr, [sp, #0]
        add sp, sp, #4
        mov pc, lr

    .data
```

```
native process 6144 In: main
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from IOExample_2...done.
(gdb) skip printf
Function printf will be skipped when stepping.
(gdb) skip scanf
Function scanf will be skipped when stepping.
(gdb) break main
Breakpoint 1 at 0x10438: file IOExample_2.s, line 8.
(gdb) run
Starting program: /home/pi/Assembly/Module4/IOExample_2

Breakpoint 1, main () at IOExample_2.s:8
(gdb) layout regs
(gdb) next
(gdb) print &name1
$1 = (<data variable, no debug info> *) 0x21060
(gdb) x/s 021060
0x2230: <error: Cannot access memory at address 0x2230>
(gdb) x/s 0x21060
0x21060:        "Chuck"
(gdb)
```

Some caveats with gdb

- It seems that names like format, name, num, etc. are all defined somewhere in gdb. Avoid label names like these. I have gotten into the habit of always appending a number to them, as in "format2", or "num1", etc. If you don't, you will not get the real addresses or be able to query them for values.

Print and x commands



```
Quit anyway? (y or n) n
Not confirmed.
(gdb) p &prompt1
$6 = (<data variable, no debug info> *) 0x2102c
(gdb) x/s 0x2102c
0x2102c:        "Enter your name: "
(gdb) 
```