GDB Tutorial

Joseph Anthony C. Hermocilla
jchermocilla@up.edu.ph

Environment

- Ubuntu 16.04 (**64-bit**)
  - gcc
  - nasm
  - binutils
  - gdb

Hello world!

```
;Save as hello_asm.asm
            extern      printf
            section     .data
msg:        db          "Hello world", 0
fmt:        db          "%s", 10, 0
            section     .text
            global      main
main:
            push        rbp
            mov         rdi,fmt
            mov         rsi,msg
            mov         rax,0
            call        printf
            mov         rbx,7
            mov         rdx,8
            pop         rbp
            mov         rax,0
            ret
```

Assemble, Link, Dump, Run

```
$nasm -f elf64 -l hello_asm.lst hello_asm.asm

$gcc -m64 -o hello_asm.exe hello_asm.o

$size hello_asm.exe hello_asm.o

$objdump -x -d -M intel hello_asm.o

$objdump -x -d -M intel hello_asm.exe

$./hello_asm.exe
```

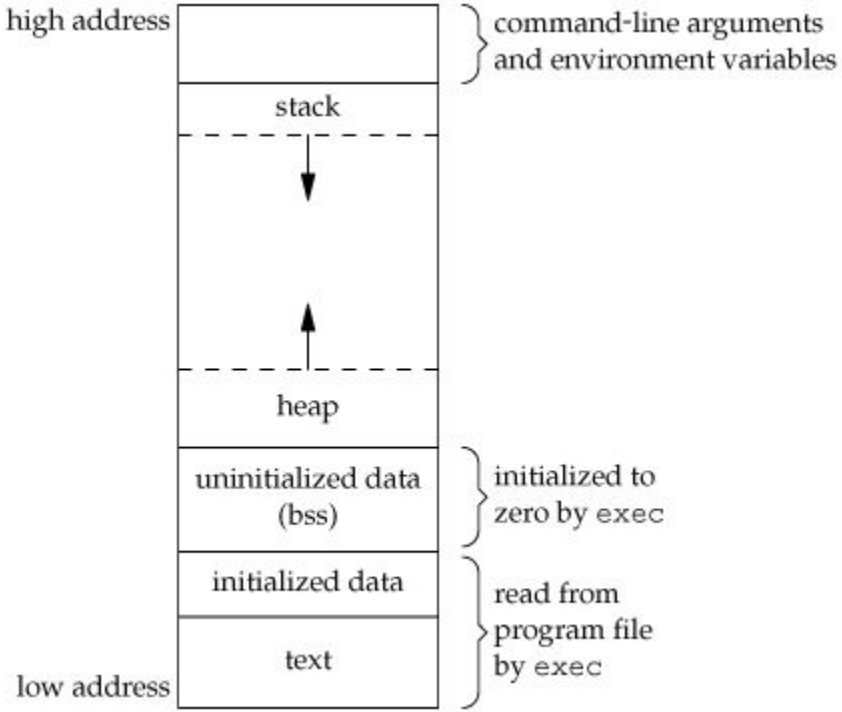
No gcc?

```
$nasm -f elf64 -l hello_asm.lst hello_asm.asm

$ld -dynamic-linker /lib64/ld-linux-x86-64.so.2
/usr/lib/x86_64-linux-gnu/crt1.o
/usr/lib/x86_64-linux-gnu/crti.o
/usr/lib/x86_64-linux-gnu/crtn.o -lc hello_asm.o -o
hello_asm.exe

$objdump -x -d -M intel hello_asm.o

$./hello_asm.exe
```

Prepare debug environment

```
$gdb hello_64.exe
(gdb) set disassembly-flavor intel
(gdb) layout asm   ;optional
(gdb) layout regs  ;optional
(gdb) disassemble main
(gdb) b main       ;set breakpoint at main
(gdb) r            ;run until breakpoint
(gdb) display/i $pc   ;display next ins
```

Aside: memory layout of a linux process



Source: http://i.stack.imgur.com/1Yz9K.gif

View mapping of program sections to memory

```
(gdb) info proc mappings
```

First line - code section (`.text`)
Second line - initialized data section (`.data`)
Third line - uninitialized data section (`.bss`)
Second to last line - stack

Where's the heap?

```
(gdb) find 0x601000, +0x1000, "Hello world"
```

In what memory area is the "Hello world" string located?

Other useful information

```
(gdb) info files
(gdb) info frame          ;call stack frame
(gdb) info stack          ;process stack info
(gdb) info proc all       ;all info about process
(gdb) info registers      ;registers
(gdb) info functions      ;functions
```

Checking the stack

```
(gdb) r                 ;run from the start
(gdb) i r rbp           ;value of bp
(gdb) i r rsp           ;value of sp
(gdb) x/4xw $rsp        ;show four words from stack
(gdb) ni                ;push rbp
(gdb) i r rsp           ;sp is updated
(gdb) x/4xw $rsp        ;value of bp now in stack
```

Calling printf

```
(gdb) ni                ;mov rdi,0x601044
(gdb) ni                ;mov rsi,0x601038
(gdb) ni                ;mov  eax,0
(gdb) ni                ;call  0x400400 <printf@plt>
```

Debug

```
$gdb hello_64.exe
(gdb) layout asm
(gdb) b main          ;set breakpoint at main
(gdb) r               ;run until breakpoint
(gdb) display/i $pc   ;display next ins
(gdb) disassemble main
(gdb) i r             ;show registers
(gdb) si              ;exec one ins
(gdb) ni              ;exec one ins (ret from fxn)
(gdb) shell clear   ;clear screen
(gdb) x/1sb 0x601044 ;examine memory string
(gdb) x/1sb 0x601038 ;examine memory string
```

How about debugging C programs?

```
//hello_c.c
#include <stdio.h>
#include <string.h>
char name[64]="Joseph";
char global[128];
int save(char *str){
    char local[128];
    strcpy(local,str);
    strcpy(global,str);
    return 0;
}
int main(){
    save("The World!");
    printf("Bye!\n");
    return 0;
}
```

Compile, Link, Dump, Run

```
$gcc -c -o hello_c.o hello_c.c

$gcc -o hello_c.exe hello_c.o

$size hello_c.exe hello_c.o

$objdump -x -d -M intel hello_c.o

$objdump -x -d -M intel hello_c.exe

$./hello_c.exe
```

Resources

- https://software.intel.com/en-us/articles/introduction-to-x64-assembly
- https://www.gnu.org/software/gdb/
-