Advanced Operating System and Virtualization

Setup Environment and read binary

Hiroaki Fukuda

Contents

- Setup Environment for minix
  - Install tools for reference
  - Linux or MaxOSX
- Read Simple Binary

Setup Environment for Minix

- Get setuptools.tar.gz

decompress

```
pine:test hiroaki$ tar -xzvf setuptools.tar.gz
```

```
x setuptools/minix2/Intel-2.0.4/i386/._NET.TAZ
x setuptools/minix2/Intel-2.0.4/i386/NET.TAZ
x setuptools/minix2/Intel-2.0.4/i386/._ROOT.MNX
x setuptools/minix2/Intel-2.0.4/i386/ROOT.MNX
x setuptools/minix2/Intel-2.0.4/i386/._USR.MNX
x setuptools/minix2/Intel-2.0.4/i386/USR.MNX
x setuptools/minix2/Intel-2.0.4/i386/._USR.TAZ
x setuptools/minix2/Intel-2.0.4/i386/USR.TAZ
```

Move to the directory

```
pine:test hiroaki$ ls
setuptools/  setuptools.tar.gz
pine:test hiroaki$ cd setuptools
pine:setuptools hiroaki$ ls
Makefile  Makefile.inc  minix2/  mmvm/  tests/
pine:setuptools hiroaki$
```

Setup Environment for Minix

Compile

```
pine:setuptools hiroaki$ make
/Applications/Xcode.app/Contents/Developer/usr/bin/make all -C mmvm
clang++  --std=c++11 -Wall -O2   -c -o binary.o binary.cpp
clang++  --std=c++11 -Wall -O2   -c -o main.o main.cpp
clang++  --std=c++11 -Wall -O2   -c -o x86disassm.o x86disassm.cpp
clang++  --std=c++11 -Wall -O2   -c -o x86dump.o x86dump.cpp
clang++  --std=c++11 -Wall -O2   -c -o dump.o dump.cpp
clang++  --std=c++11 -Wall -O2   -c -o v6dump.o v6dump.cpp
clang++  --std=c++11 -Wall -O2   -c -o x86runtime.o x86runtime.cpp
clang++  --std=c++11 -Wall -O2   -c -o unixv6.o unixv6.cpp
clang++  --std=c++11 -Wall -O2   -c -o x86unixv6.o x86unixv6.cpp
clang++  --std=c++11 -Wall -O2   -c -o minix.o minix.cpp
clang++  --std=c++11 -Wall -O2   -c -o x86minix.o x86minix.cpp
clang++  --std=c++11 -Wall -O2   -c -o os.o os.cpp
clang++  --std=c++11 -Wall -O2 -o mmvm binary.o main.o x86disassm.o x86dump.o dump.o v6dum
p.o x86runtime.o unixv6.o x86unixv6.o minix.o x86minix.o os.o
pine:setuptools hiroaki$
```

Install

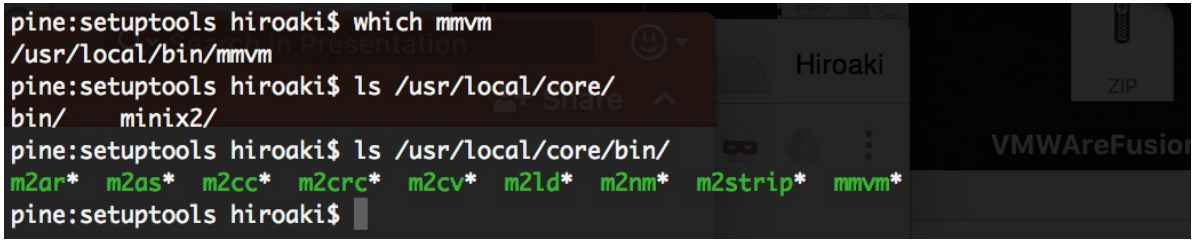
```
pine:setuptools hiroaki$ sudo make install
Password:
```

Setup

```
hiroaki@pine setuptools % sudo make setup
Password:
```

Verify the installation

Find “mmvm” and “m2XX” in each directory



Test the tools

Compile and execute a simple program

```
pine:setuptools hiroaki$ ls
Makefile  Makefile.inc  minix2/  mmvm/  tests/
pine:setuptools hiroaki$ cd tests/
pine:tests hiroaki$ ls
1.c  2.c  3.c  4.c  5.c  6.c  asem/  module.c
pine:tests hiroaki$ cd asem/
pine:asem hiroaki$ /usr/local/core/bin/m2cc -.o 1.s
pine:asem hiroaki$ ls
1.s  2.s  3.s  a.out*
pine:asem hiroaki$ mmvm a.out
hello
```

Execute with logs

```
pine:asem hiroaki$ mmvm -m a.out
 AX   BX   CX   DX   SP   BP   SI   DI  FLAGS IP
0000 0000 0000 0000 ffdc 0000 0000 0000 ---- 0000:bb0000       mov bx, 0000
0000 0000 0000 0000 ffdc 0000 0000 0000 ---- 0003:cd20         int 20
<write(1, 0x0020, 6)hello
 => 6>
0000 0000 0000 0000 ffdc 0000 0000 0000 ---- 0005:bb1000       mov bx, 0010
0000 0010 0000 0000 ffdc 0000 0000 0000 ---- 0008:cd20         int 20
<exit(0)>
```

Disassemble

```
pine:asem hiroaki$ mmvm -d a.out
0000: bb0000        mov bx, 0000
0003: cd20          int 20
0005: bb1000        mov bx, 0010
0008: cd20          int 20
000a: 0000          add [bx+si], al
000c: 0000          add [bx+si], al
000e: 0000          add [bx+si], al
```

If you can see everything is the same, Congrats!!

Try another examples

Read simple binary

Read a.out using hexdump

```
pine:asem hiroaki$ hexdump -C a.out
00000000  01 03 20 04 20 00 00 00  10 00 00 00 26 00 00 00  |.. . .......&...|
00000010  00 00 00 00 00 00 00 00  00 00 01 00 70 00 00 00  |............p...|
00000020  bb 00 00 cd 20 bb 10 00  cd 20 00 00 00 00 00 00  |.... .... ......|
00000030  01 00 04 00 01 00 06 00  00 00 20 00 00 00 00 00  |.......... .....|
00000040  01 00 01 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  68 65 6c 6c 6f 0a 6d 65  73 73 61 67 65 00 00 00  |hello.message...|
00000060  00 00 03 00 00 00 65 78  69 74 00 00 00 00 10 00  |......exit......|
00000070  00 00 03 00 00 00 68 65  6c 6c 6f 00 00 00 20 00  |......hello... .|
00000080  00 00 03 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 02 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000a0  00 00 03 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 03 00 00 00 00 00  00 00 00 00 00 00 26 00  |..............&.|
000000c0  00 00 04 00 00 00                                 |......|
000000c6
```

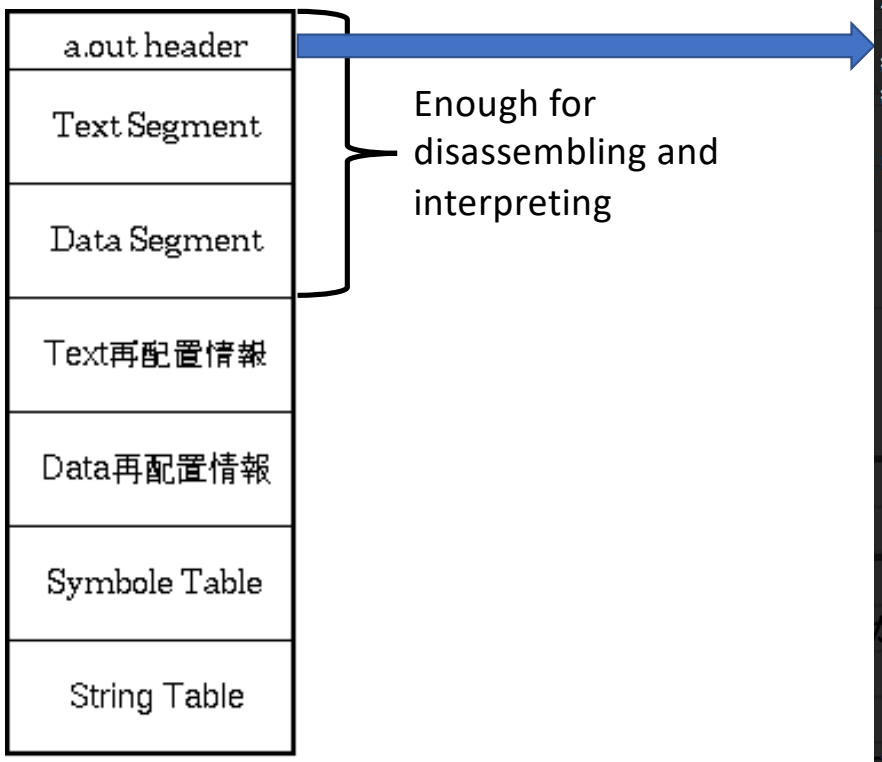
Disassemble

```
pine:asem hiroaki$ mmvm -d a.out
0000: bb0000        mov bx, 0000
0003: cd20          int 20
0005: bb1000        mov bx, 0010
0008: cd20          int 20
000a: 0000          add [bx+si], al
000c: 0000          add [bx+si], al
000e: 0000          add [bx+si], al
```

Why??

- a.out is a format to make a structure of a program
- Disassembler only disassemble its program(text) part

a.out format



usr/include/a.out.h

```
/* The <a.out> header file describes the format of executable files. */

#ifndef _AOUT_H
#define _AOUT_H

struct  exec {                  /* a.out header */
  unsigned char a_magic[2];     /* magic number */
  unsigned char a_flags;        /* flags, see below */
  unsigned char a_cpu;          /* cpu id */
  unsigned char a_hdrlen;       /* length of header */
  unsigned char a_unused;       /* reserved for future use */
  unsigned short a_version;     /* version stamp (not used at present) */
  long          a_text;         /* size of text segement in bytes */
  long          a_data;         /* size of data segment in bytes */
  long          a_bss;          /* size of bss segment in bytes */
  long          a_entry;        /* entry point */
  long          a_total;        /* total memory allocated */
  long          a_syms;         /* size of symbol table */

  /* SHORT FORM ENDS HERE */
  long          a_trsize;       /* text relocation size */
  long          a_drsize;       /* data relocation size */
  long          a_tbase;        /* text relocation base */
  long          a_dbase;        /* data relocation base */
};
```

Binary Again

```
pine:asem hiroaki$ mmvm -d a.out
0000: bb0000        mov bx, 0000
0003: cd20          int 20
0005: bb1000        mov bx, 0010
0008: cd20          int 20
000a: 0000          add [bx+si], al
000c: 0000          add [bx+si], al
000e: 0000          add [bx+si], al
```

Little Endian

0x00000010 = 16

```
pine:asem hiroaki$ hexdump -C a.out
00000000  01 03 20 04 20 00 00 00  10 00 00 00 26 00 00 00  |.. . .......&...|
00000010  00 00 00 00 00 00 00 00  00 00 01 00 70 00 00 00  |............p...|
00000020  bb 00 00 cd 20 bb 10 00  cd 20 00 00 00 00 00 00  |.... .... ......|
00000030  01 00 04 00 01 00 06 00  00 00 20 00 00 00 00 00  |.......... .....|
00000040  01 00 01 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  68 65 6c 6c 6f 0a 6d 65  73 73 61 67 65 00 00 00  |hello.message...|
00000060  00 00 03 00 00 00 65 78  69 74 00 00 00 00 10 00  |......exit......|
00000070  00 00 03 00 00 00 68 65  6c 6c 6f 00 00 00 20 00  |......hello... .|
00000080  00 00 03 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 02 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000a0  00 00 03 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 03 00 00 00 00 00  00 00 00 00 00 00 26 00  |..............&.|
000000c0  00 00 04 00 00 00                                 |......|
000000c6
```

Let’s get started to implement disassembler

Focus on 1.s

1. Read binary (e.g., a.out)
2. Analyze header
3. Extract text area
4. Display as ”mmvm –d a.out”

Don’t think why we see output like that!

Next week… you can find it