CSGE602055 Operating Systems
CSF2600505 Sistem Operasi
Minggu 04: Addressing, Shared Lib, Pointer & I/O Programming

Rahmat M. Samik-Ibrahim

Universitas Indonesia

http://rms46.vlsm.org/2/207.html

REV79 02-Oct-2017

Jadwal OS172

Minggu 00	29 Aug - 05 Sep 2017	Intro & Review
Minggu 01	07 Sep - 12 Sep 2017	IPR, SED, AWK, REGEX, & Scripting
Minggu 02	14 Sep - 19 Sep 2017	Protection, Security, Privacy, & C-language
Minggu 03	26 Sep - 30 Sep 2017	BIOS, Loader, Systemd, & I/O
Minggu 04	03 Okt - 07 Okt 2017	Addressing, Shared Lib, Pointer & I/O Programming
Minggu 05	10 Okt - 14 Okt 2017	Virtual Memory
Ming. UTS	15 Okt - 24 Okt 2017	
Minggu 06	26 Okt - 31 Okt 2017	Concurency: Processes & Threads
Minggu 07	02 Nov - 07 Nov 2017	Synchronization
Minggu 08	09 Nov - 14 Nov 2017	Scheduling & Network Sockets Programming
Minggu 09	16 Nov - 21 Nov 2017	File System & Persistent Storage
Minggu 10	23 Nov - 28 Nov 2017	Special Topic: Blockchain
Cadangan	30 Nov - 09 Des 2017	
Ming. UAS	10 Des - 23 Des 2017	

Agenda

1. Start
2. Agenda
3. Week 04
4. Addressing
5. Global Variables
6. Local Variables
7. Pointers
8. Pointers of Pointers
9. Pointers of Pointers of Pointers
10. Character Pointer vs Integer Pointer
11. The End

Week 04: Addressing, Shared Lib, Pointer & I/O Prog

- Reference (I/O): (OLD 08)
- 8 bit Variable (eg. `int ii=10;`)
  - Value (10_{10} == `0x 0A`)
  - Logical Address (eg. `0x 0040`)
  - Meaning & Context (Variabel `"ii"` is an integer).
  - `[0x 0040] == 0x 0A`
- Multiple Address Variable ($>$ 1 byte size)
  - Little-Endian (LE)
  - Big-Endian (BE)
  - Bi-Endian
- Executable File Format
  - Ancient Linux/Unix: Assembler Output $\rightarrow$ [a.out].
  - iOS, MacOS: Mach-Output (Mach-O).
  - Linux: Executable and Linking Format (ELF).
  - Windows: Portable Executable (PE) $\rightarrow$ [.acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp].

Addressing (Eg. 16 bits)

16 Bits Logical Address Table (HEX)																Examples			
ADDR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	bits	L/B	PTR	VALUE
000X	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	8	—	**[0008]**	**A8**
001X	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	8	—	**[0014]**	**B4**
002X	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	8	—	**[0015]**	**B5**
003X	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	16	LE	**[0014]**	**B5 B4**
004X	0A																16	BE	**[0014]**	**B4 B5**
⋮									⋮							⋮	32	LE	**[0014]**	**B7 B6 B5 B4**
FFFX																	1 address == 1 byte LE: Little Endian BE: Big Endian			

Global Variables

```
/* Global Variables in Data Segment*/
char   varchr0='a';
char   varchr1='b';
char   varchr2='c';
char   varchr3='d';
char   varchr4='e';
char   varchr5='f';
char   varchr6='g';
char   varchr7='h';
=======================================
varchr0: value =  a, address = 0x601038
varchr1: value =  b, address = 0x601039
varchr2: value =  c, address = 0x60103a
varchr3: value =  d, address = 0x60103b
varchr4: value =  e, address = 0x60103c
varchr5: value =  f, address = 0x60103d
varchr6: value =  g, address = 0x60103e
varchr7: value =  h, address = 0x60103f
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60103X									'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'

Memory Map

```
Name            Origin              Length              Attributes
*default*       0x0000000000000000 0xffffffffffffffff
                                                        PLT=Procedure Linkage
    Table
.plt            0x0000000000400420  0x30                /usr/lib/..../crt1.o
                0x0000000000400430                      puts@@GLIBC\_2.2.5
                0x0000000000400440                      printf@@GLIBC\_2.2.5

.text           0x0000000000400450 0x282

.data           0x0000000000601028  0x18
.data           0x0000000000601038   0x8                /tmp/cc0DQ6w0.o
                0x0000000000601038                      varchr0
                0x0000000000601039                      varchr1
                ...                                     ...
                0x000000000060103e                      varchr6
                0x000000000060103f                      varchr7

.bss            0x0000000000601040   0x8
```

Local Variables

```
/* Local Variables in Stack Segment */
char   varchr0='a';
char   varchr1='b';
char   varchr2='c';
char   varchr3='d';
char   varchr4='e';
char   varchr5='f';
char   varchr6='g';
char   varchr7='h';
==========================================
varchr0: value =  a, address = 0x7ffd70c4facf
varchr1: value =  b, address = 0x7ffd70c4face
varchr2: value =  c, address = 0x7ffd70c4facd
varchr3: value =  d, address = 0x7ffd70c4facc
varchr4: value =  e, address = 0x7ffd70c4facb
varchr5: value =  f, address = 0x7ffd70c4faca
varchr6: value =  g, address = 0x7ffd70c4fac9
varchr7: value =  h, address = 0x7ffd70c4fac8
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00007ffd–70c4facX									'h'	'g'	'f'	'e'	'd'	'c'	'b'	'a'

Pointers

```
/* Global Variables in Data Segment*/
char   varchr0='a';
char   varchr1='b';
char   varchr2='c';
char   varchr3='d';
char*  ptrchr0=&varchr0;
char*  ptrchr1=&varchr1;
char*  ptrchr2=&varchr2;
char*  ptrchr3=&varchr3;
============================================================
varchr0: value =   a, address = 0x601038
varchr1: value =   b, address = 0x601039
varchr2: value =   c, address = 0x60103a
varchr3: value =   d, address = 0x60103b
ptrchr0: points to a, value   = 0x601038,  address = 0x601040
ptrchr1: points to b, value   = 0x601039,  address = 0x601048
ptrchr2: points to c, value   = 0x60103a,  address = 0x601050
ptrchr3: points to d, value   = 0x60103b,  address = 0x601058
============================================================
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000–0060103X									'a'	'b'	'c'	'd'				
00000000–0060104X	00000000–00601038								00000000–00601039							
00000000–0060105X	3A	10	60	00	00	00	00	00	3B	10	60	00	00	00	00	00

Pointers of Pointers

```
/* Global Variables in Data Segment*/
char   varchr0='a';
char   varchr1='b';
char   varchr2='c';
char   varchr3='d';
char*  ptrchr0=&varchr0;
char*  ptrchr1=&varchr1;
char*  ptrchr2=&varchr2;
char*  ptrchr3=&varchr3;
char** ptrptr0=&ptrchr0;
char** ptrptr1=&ptrchr1;
char** ptrptr2=&ptrchr2;
char** ptrptr3=&ptrchr3;
=====================================================
varchr0: val=a, adr=0x601038
varchr1: val=b, adr=0x601039
varchr2: val=c, adr=0x60103a
varchr3: val=d, adr=0x60103b
ptrchr0: pts=a, val=0x601038, adr=0x601040
ptrchr1: pts=b, val=0x601039, adr=0x601048
ptrchr2: pts=c, val=0x60103a, adr=0x601050
ptrchr3: pts=d, val=0x60103b, adr=0x601058
ptrptr0: ppt=a, pts=0x601038, val=0x601040, adr=0x601060
ptrptr1: ppt=b, pts=0x601039, val=0x601048, adr=0x601068
ptrptr2: ppt=c, pts=0x60103a, val=0x601050, adr=0x601070
ptrptr3: ppt=d, pts=0x60103b, val=0x601058, adr=0x601078
=====================================================
```

Pointers of Pointers (2)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60103X									'a'	'b'	'c'	'd'				
60104X	601038								601039							
60105X	60103A								60103B							
60106X	601040								601048							
60107X	601050								601058							

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000–0060103X									61	62	63	64				
00000000–0060104X	38	10	60	00	00	00	00	00	39	10	60	00	00	00	00	00
00000000–0060105X	3A	10	60	00	00	00	00	00	3B	10	60	00	00	00	00	00
00000000–0060106X	40	10	60	00	00	00	00	00	48	10	60	00	00	00	00	00
00000000–0060107X	50	10	60	00	00	00	00	00	58	10	60	00	00	00	00	00

Pointers of Pointers of Pointers

```
/* Global Variables in Data Segment*/
char   varchr0='a';
char   varchr1='b';
char   varchr2='c';
char   varchr3='d';
char*  ptrchr0=&varchr0;
char*  ptrchr1=&varchr1;
char*  ptrchr2=&varchr2;
char*  ptrchr3=&varchr3;
char** ptrptr0=&ptrchr0;
char** ptrptr1=&ptrchr1;
char** ptrptr2=&ptrchr2;
char** ptrptr3=&ptrchr3;
char*** ppptr0=&ptrptr0;
=====================================================================
varchr0: val=a, adr=0x601038
varchr1: val=b, adr=0x601039
varchr2: val=c, adr=0x60103a
varchr3: val=d, adr=0x60103b
ptrchr0: pts=a, val=0x601038, adr=0x601040
ptrchr1: pts=b, val=0x601039, adr=0x601048
ptrchr2: pts=c, val=0x60103a, adr=0x601050
ptrchr3: pts=d, val=0x60103b, adr=0x601058
ptrptr0: ppt=a, pts=0x601038, val=0x601040, adr=0x601060
ptrptr1: ppt=b, pts=0x601039, val=0x601048, adr=0x601068
ptrptr2: ppt=c, pts=0x60103a, val=0x601050, adr=0x601070
ptrptr3: ppt=d, pts=0x60103b, val=0x601058, adr=0x601078
ppptr0:  ppp=a, ppt=0x601038, pts=0x601040, val=0x601060, adr=0x601080
=====================================================================
```

Pointers of Pointers of Pointer (2)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60103X									'a'	'b'	'c'	'd'				
60104X	601038								601039							
60105X	60103A								60103B							
60106X	601040								601048							
60107X	601050								601058							
60108X	601060															

- `***ppptr0 = **ptrptr0 = *ptrchr = varchr0`
- `ppptr0  = [601080] = 601060`
- `ptrptr0 = [601060] = 601040`
- `ptrchr0 = [601040] = 601038`
- `varchr0 = [601038] = 'a'`

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000–0060103X									61	62	63	64				
00000000–0060104X	38	10	60	00	00	00	00	00	39	10	60	00	00	00	00	00
00000000–0060105X	3A	10	60	00	00	00	00	00	3B	10	60	00	00	00	00	00
00000000–0060106X	40	10	60	00	00	00	00	00	48	10	60	00	00	00	00	00
00000000–0060107X	50	10	60	00	00	00	00	00	58	10	60	00	00	00	00	00
00000000–0060108X	60	10	60	00	00	00	00	00								

Character Pointer vs Integer Pointer

```
=======================================
/* Global Variables in Data Segment*/
int     varint0=0x41424344;
char    varchr0='a';
char    varchr1='b';
char    varchr2='c';
char    varchr3='d';

int*    ptrint0=&varint0;
char*   ptrchr0=&varchr0;

ptrint0=(int*) &varchr2;
varint0=*ptrint0;

ptrchr0=(char*) &varint0;
varchr0=*ptrchr0;

ptrchr0++;
varchr0=*ptrchr0;
=======================================
```

Character Pointer vs Integer Pointer (2)

```
Value of: ==============
varint0 = 0X41424344 = D
varchr0 =       0X61 = a
varchr1 =       0X62 = b
varchr2 =       0X63 = c
varchr3 =       0X64 = d
ptrint0 =   0x601038
ptrchr0 =   0x60103c
Address of: ========
varint0 =   0x601038
varchr0 =   0x60103c
varchr1 =   0x60103d
varchr2 =   0x60103e
varchr3 =   0x60103f
ptrint0 =   0x601040
ptrchr0 =   0x601048
Points to: =========
ptrint0 = 0X41424344
ptrchr0 =          a
ptrint0=(int*) &varchr2; ========
varint0=*ptrint0; ==============
```

Character Pointer vs Integer Pointer (3)

```
Value of: ==============
ptrint0 =   0x60103e
varint0 = 0X103E6463 = c
Address of: ========
ptrint0 =   0x601040
varint0 =   0x601038
ptrchr0=(char*) &varint0; =======
varchr0=*ptrchr0; ===============
Value of: ==============
ptrchr0 =   0x601038
varchr0 =       0X63 = c
Address of: ========
ptrchr0 =   0x601048
varchr0 =   0x60103c
ptrchr0++; ======================
Value of: ==============
ptrchr0 =   0x601039
varchr0 =       0X64 = d
Address of: ========
ptrchr0 =   0x601048
varchr0 =   0x60103c
```

The End

- This is the end of the presentation.