

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>

REV083 05-Oct-2017

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	Concurrency: 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 I

- 1 Start
- 2 Agenda
- 3 Week 04
- 4 Addressing
- 5 Makefile
- 6 00-global-variables
- 7 Linux Libraries
- 8 01-local-variables
- 9 02-pointers
- 10 03-pointers-of-pointers
- 11 04-pointers-of-pointers-of-pointers
- 12 05-chrptr-vs-intptr
- 13 06-pointer-address
- 14 07-addresses
- 15 08-passing-parameters
- 16 09-struct

Agenda II

- 17 50-get-put — 51-get-put-loop
- 18 52-open-close
- 19 53-file-pointer
- 20 54-write
- 21 55-write
- 22 56-copy
- 23 57-dup
- 24 58-dup2
- 25 59-io
- 26 60-readwrite
- 27 The End

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

- Reference (I/O): (OLD 08)
- This will be a difficult week
 - Pray! Pray! We got to pray just to make it today (McH)!
 - Goosfraba: Turn To Page 394 (AM-HP3)!
- 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 → `[a.out]`.
 - iOS, MacOS: Mach-Output (Mach-O).
 - Linux: Executable and Linking Format (ELF).
 - Windows: Portable Executable (PE) →
`[.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			

Makefile

```
CC=gcc
P00=00-global-variables
P01=01-local-variables
...

EXECS= \
    $(P00) \
    $(P01) \
...

DEMOFILES=\
    demo-file1.txt \
    demo-file2.txt \
...

all: $(EXECS)

$(P00): $(P00).c
    $(CC) $(P00).c -o $(P00) -Xlinker -Map=$(P00).map

$(P01): $(P01).c
    $(CC) $(P01).c -o $(P01) -Xlinker -Map=$(P01).map
...

$(P04): $(P04).c
    $(CC) $(P04).c -o $(P04)
...
clean:
    rm -f ${EXECS}
...
demo:
    bash .shsh
```

00-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';
```

```
VARIABLE  +++  VALUE  +CHR+  + ADDRESS+
```

```
varchr0 =          0X61 = a      0x601038
```

```
varchr1 =          0X62 = b      0x601039
```

```
varchr2 =          0X63 = c      0x60103a
```

```
varchr3 =          0X64 = d      0x60103b
```

```
varchr4 =          0X65 = e      0x60103c
```

```
varchr5 =          0X66 = f      0x60103d
```

```
varchr6 =          0X67 = g      0x60103e
```

```
varchr7 =          0X68 = h      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

Memory Configuration (00-global-char.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	

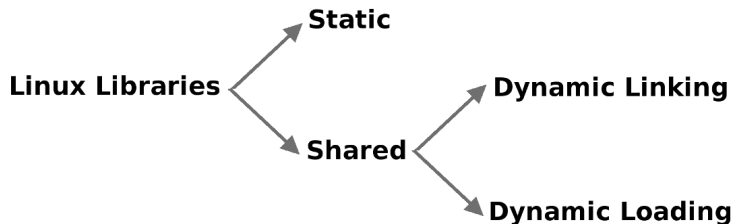


Figure: Linux Libraries

- Static Libraries (embedded in the program).
 - Self contained
 - StaticLib.a
- Shared Libraries
 - Dynamic Linking (run-time.so).
 - Dynamic Loading (controlled by the program, DL-API).

01-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';
```

```
VARIABLE  +++  VALUE  +CHR+  +++  ADDRESS  +++
```

```
varchr0 =          0X61 = a      0x7ffcc188b51f
```

```
varchr1 =          0X62 = b      0x7ffcc188b51e
```

```
varchr2 =          0X63 = c      0x7ffcc188b51d
```

```
varchr3 =          0X64 = d      0x7ffcc188b51c
```

```
varchr4 =          0X65 = e      0x7ffcc188b51b
```

```
varchr5 =          0X66 = f      0x7ffcc188b51a
```

```
varchr6 =          0X67 = g      0x7ffcc188b519
```

```
varchr7 =          0X68 = h      0x7ffcc188b518
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00007ffc-c188b51X									'h'	'g'	'f'	'e'	'd'	'c'	'b'	'a'

02-pointers (LE: Little Endian)

```
char   varchr0='a';
char   varchr1='b';
char   varchr2='c';
char   varchr3='d';
char*  ptrchr0=&varchr0;
char*  ptrchr1=&varchr1;
char*  ptrchr2=&varchr2;
char*  ptrchr3=&varchr3;
```

VARIABLE	+++	VALUE	+CHR+	+ADDRESS	+POINTS TO+
varchr0	=	0X61	= a	0x601038	
varchr1	=	0X62	= b	0x601039	
varchr2	=	0X63	= c	0x60103a	
varchr3	=	0X64	= d	0x60103b	
ptrchr0	=	0x601038		0x601040	a
ptrchr1	=	0x601039		0x601048	b
ptrchr2	=	0x60103a		0x601050	c
ptrchr3	=	0x60103b		0x601058	d

	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

03-pointers-of-pointers (LE)

```
=====
/* 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;
VARIABLE  +++  VALUE +CHR+ +ADDRESS + +POINTS TO+
varchr0 =      0x61 = a      0x601038
varchr1 =      0x62 = b      0x601039
varchr2 =      0x63 = c      0x60103a
varchr3 =      0x64 = d      0x60103b
ptrchr0 = 0x601038      0x601040      a
ptrchr1 = 0x601039      0x601048      b
ptrchr2 = 0x60103a      0x601050      c
ptrchr3 = 0x60103b      0x601058      d
ptrptr0 = 0x601040      0x601060 0x601038
ptrptr1 = 0x601048      0x601068 0x601039
ptrptr2 = 0x601050      0x601070 0x60103a
ptrptr3 = 0x601058      0x601078 0x60103b
=====
```

03-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

04-pointers-of-pointers-of-pointers (LE)

```
/* 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;
```

VARIABLE	+++	VALUE	+CHR+	+ADDRESS +	+POINTS TO+
varchr0	=	0X61	= a	0x601038	
varchr1	=	0X62	= b	0x601039	
varchr2	=	0X63	= c	0x60103a	
varchr3	=	0X64	= d	0x60103b	
ptrchr0	=	0x601038		0x601040	a
ptrchr1	=	0x601039		0x601048	b
ptrchr2	=	0x60103a		0x601050	c
ptrchr3	=	0x60103b		0x601058	d
ptrptr0	=	0x601040		0x601060	0x601038
ptrptr1	=	0x601048		0x601068	0x601039
ptrptr2	=	0x601050		0x601070	0x60103a
ptrptr3	=	0x601058		0x601078	0x60103b
ppptr0	=	0x601060		0x601080	0x601040

04-pointers-of-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							
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								

05-chrptr-vs-intptr (LE)

```
=====
/* 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;
=====
```

05-chrptr-vs-intptr (2)

```
VARIABLE  +++  VALUE +CHR+ +ADDRESS + +POINTS TO+++  
varint0 = 0X41424344 = D      0x601038  
varchr0 =           0X61 = a      0x60103c  
varchr1 =           0X62 = b      0x60103d  
varchr2 =           0X63 = c      0x60103e  
varchr3 =           0X64 = d      0x60103f  
ptrint0 = 0x601038           0x601048  0X41424344  
ptrchr0 = 0x60103c           0x601050      a  
!!! ptrint0=(int*) &varchr1;  varint0=*ptrint0; !!!  
VARIABLE  +++  VALUE +CHR+ +ADDRESS + +POINTS TO+++  
ptrint0 = 0x60103d           0x601048  0X65646362  
varint0 = 0X65646362 = b      0x601038
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000-0060103X									44	43	42	41	61	62	63	64
00000000-0060104X	65								38	10	60	00	00	00	00	00
00000000-0060105X	3C	10	60	00	00	00	00	00								

00000000-0060103X									62	63	64	65	61	62	63	64
00000000-0060104X	65								3D	10	60	00	00	00	00	00

05-chrptr-vs-intptr (2)

```
!!! ptrchr0=(char*) &varint0; varchr0=*ptrchr0; !!!  
VARIABLE  +++  VALUE +CHR+ +ADDRESS + +POINTS TO+++  
ptrchr0 =    0x601038          0x601050          0X62  
varchr0 =          0X62 = b    0x60103c  
!!!! !!!!! ptrchr0++; varchr0=*ptrchr0; !!!!! !!!!!  
VARIABLE  +++  VALUE +CHR+ +ADDRESS + +POINTS TO+++  
ptrchr0 =    0x601039          0x601050          0X63  
varchr0 =          0X63 = c    0x60103c
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000-0060103X									44	43	42	41	61	62	63	64
00000000-0060104X	65								38	10	60	00	00	00	00	00
00000000-0060105X	3C	10	60	00	00	00	00	00								
00000000-0060103X									62	63	64	65	61	62	63	64
00000000-0060104X	65								3D	10	60	00	00	00	00	00
00000000-0060103X									62	63	64	65	62	62	63	64
00000000-0060105X	38	10	60	00	00	00	00	00								
00000000-0060103X									62	63	64	65	63	62	63	64
00000000-0060105X	39	10	60	00	00	00	00	00								

06-pointer-address (LE)

```
unsigned char   varchr0='a';
unsigned char*  ptrchr0=&varchr0;
unsigned char*  ptrcopy=(char *) &ptrchr0;
```

VARIABLE	+++	VALUE	+++	+CHR+	+++	ADDRESS	+++	+PTS	TO+
varchr0 =		0X61	= a			0x7ffe7bb7369f			
ptrchr0 =	0x7ffe7bb7369f					0x7ffe7bb73690		0X61	

```
!!! !!!!! ptrcopy++; ptrcopy++; ptrcopy++; ... !!!!! !!!
ptrcopy = 0x7ffe7bb73690      0x7ffe7bb73688      0X9F
ptrcopy = 0x7ffe7bb73691      0x7ffe7bb73688      0X36
ptrcopy = 0x7ffe7bb73692      0x7ffe7bb73688      0XB7
ptrcopy = 0x7ffe7bb73693      0x7ffe7bb73688      0X7B
ptrcopy = 0x7ffe7bb73694      0x7ffe7bb73688      0XFE
ptrcopy = 0x7ffe7bb73695      0x7ffe7bb73688      0X7F
ptrcopy = 0x7ffe7bb73696      0x7ffe7bb73688      00
ptrcopy = 0x7ffe7bb73697      0x7ffe7bb73688      00
```

06-pointer-address (2)

```

!!! !!!!! ptrcopy++; ptrcopy++; ptrcopy++; ... !!!!! !!!
VARIABLE  +++  VALUE  +++ +CHR+  +++ ADDRESS  +++ +PTS TO+
ptrchr0 = 0x7ffe7bb7369f          0x7ffe7bb73690      0X61
ptrcopy = 0x7ffe7bb73690          0x7ffe7bb73688      0X9F
ptrcopy = 0x7ffe7bb73691          0x7ffe7bb73688      0X36
ptrcopy = 0x7ffe7bb73692          0x7ffe7bb73688      0XB7
ptrcopy = 0x7ffe7bb73693          0x7ffe7bb73688      0X7B
ptrcopy = 0x7ffe7bb73694          0x7ffe7bb73688      0XFE
ptrcopy = 0x7ffe7bb73695          0x7ffe7bb73688      0X7F
ptrcopy = 0x7ffe7bb73696          0x7ffe7bb73688        00
ptrcopy = 0x7ffe7bb73697          0x7ffe7bb73688        00

```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00007FFE-7BB7368X									90	36	B7	7B	FE	7F	00	00
00007FFE-7BB7369X	9F	36	B7	7B	FE	7F	00	00								61
00007FFE-7BB7368X									91	36	B7	7B	FE	7F	00	00
00007FFE-7BB7368X									92	36	B7	7B	FE	7F	00	00
00007FFE-7BB7368X									93	36	B7	7B	FE	7F	00	00
00007FFE-7BB7368X									94	36	B7	7B	FE	7F	00	00
00007FFE-7BB7368X									95	36	B7	7B	FE	7F	00	00
00007FFE-7BB7368X									96	36	B7	7B	FE	7F	00	00
00007FFE-7BB7368X									97	36	B7	7B	FE	7F	00	00

07-addresses (LE)

```
unsigned int  glInt1 = 0x41;
unsigned int  glInt2 = 0x42;
unsigned int  glInt3 = 0x43;
unsigned int  glInt4 = 0x44;
unsigned int  glInt5 = 0x45;
unsigned int* heapArray[] =
    {&glInt1, &glInt2, &glInt3, &glInt4, &glInt5};
```

Variable Name	Address	Size(S)/Value(V)
=====		
glInt1	0x601060	0X41 (V)
glInt2	0x601064	0X42 (V)
glInt3	0x601068	0X43 (V)
glInt4	0x60106c	0X44 (V)
heapArray---	0x601080	0X601060 (V)
heapArray[0]	0x601080	0X601060 (V)
heapArray[1]	0x601088	0X601064 (V)
heapArray[2]	0x601090	0X601068 (V)
heapArray[3]	0x601098	0X60106C (V)
heapArray[4]	0x6010a0	0X601070 (V)

07-addresses (2)

```
#define ALLOC0 0x4BD8
#define ALLOC1 0xFF8
#define ALLOC2 0x18
#define ALLOC3 0x19
#define ALLOC4 1
heapArray[0]=malloc(ALLOC0);
heapArray[1]=malloc(ALLOC1);
heapArray[2]=malloc(ALLOC2);
heapArray[3]=malloc(ALLOC3);
heapArray[4]=malloc(ALLOC4);
```

Variable Name	Address	Size(S)/Value(V)
=====		
heapArray---	0x601080	0X23CF420 (V)
heapArray[0]	0x601080	0X23CF420 (V)
heapArray[1]	0x601088	0X23D4000 (V)
heapArray[2]	0x601090	0X23D5000 (V)
heapArray[3]	0x601098	0X23D5020 (V)
heapArray[4]	0x6010a0	0X23D5050 (V)

07-addresses (3)

```
long printVariable(char* varName, void* varValue, long endAddr) { ... }
long printHeapArray(int mode) { ... }
long demoMalloc(int mode) { ... }
long tripleLoop(int mode) { ... }
void main(void)          { ... }
```

Variable Name	Address	Size(S)/Value(V)
printf	0x400480	
malloc	0x400490	
printVariable	0x400596	0XBE (S)
printHeapArray	0x400654	0XA3 (S)
demoMalloc	0x4006f7	0X7E (S)
tripleLoop	0x400775	0XFC (S)
main	0x400871	0X148 (S)

07-addresses (3)

#####

Memory Configuration

	0x0000000000400238	(SEGMENT-START ("text-segment", 0x400000) + SIZEOF-HEADERS)
.plt	0x0000000000400460	0x40 /usr/lib/gcc/.../x86-64-linux-gnu/crt1.o
	0x0000000000400470	puts@@GLIBC_2.2.5
	0x0000000000400480	printf@@GLIBC_2.2.5
	0x0000000000400490	malloc@@GLIBC_2.2.5
.text	0x00000000004004a0	0x592
.text	0x0000000000400596	0x41d /tmp/ccU78N7D.o
	0x0000000000400596	printVariable
	0x0000000000400654	printHeapArray
	0x00000000004006f7	demoMalloc
	0x0000000000400775	tripleLoop
	0x0000000000400871	main
.data	0x0000000000601060	0x48 /tmp/ccU78N7D.o
	0x0000000000601060	glInt1
	0x0000000000601064	glInt2
	0x0000000000601068	glInt3
	0x000000000060106c	glInt4
	0x0000000000601070	glInt5
	0x0000000000601080	heapArray

#####

08-passing-parameters

```
#define NOP()    __asm__ ("nop") /* No Operation inline gcc ASM *** */
#include <stdio.h>
int  varInt1    = 0x01;
int  varInt2    = 0x02;
int* ptrInt1    = &varInt1;
int* ptrInt2    = &varInt2;
void function1(void) {
    NOP();
}
void function2(int iif2) {
    printf("function2:    iif2 = %d\n", ++iif2);
}
void function3(int* iif3) {
    printf("function3:    iif3 = %d\n", ++(*iif3));
}
int  function4(void) {
    NOP();
}
int* function5(void) {
    NOP();
}
void main(void) {
    function1();
    printf("main-1:    *ptrInt1 = %d\n", *ptrInt1);
    function2(*ptrInt1);
    printf("main-2:    *ptrInt1 = %d\n", *ptrInt1);
    printf("main-3:    varInt1 = %d\n",  varInt1);
    function3(&varInt1);
    printf("main-4:    varInt1 = %d\n",  varInt1);
}
```

*// main-1: *ptrInt1 = 1*
// function2: iif2 = 2
*// main-2: *ptrInt1 = 1*
// main-3: varInt1 = 1
// function3: iif3 = 2
// main-4: varInt1 = 2

09-struct

```
#include <stdio.h>

typedef struct {
    char* nama;
    int umur;
    int semester;
    char* NIM;
} student;

void printStruct(student* ss) {
    printf("%-10s %11s %3d %2d\n", ss->nama, ss->NIM, ss->umur, ss->semester);
}

student global;
void init(void) {
    global.nama      = "Burhan";
    global.NIM       = "1205000003";
    global.umur      = 10;
    global.semester  = 2;
}

void main(void) {
    student mhs = {"Ali", 12, 1, "1205000001"};
    printStruct(&mhs);
    init();
    printStruct(&global);
}

=====
Ali          1205000001  12  1
Burhan       1205000003  10  2
```

50-get-put — 51-get-put-loop

```
#include <stdio.h>
```

```
void main (void) {  
    int cc = getchar();  
    putchar(cc);  
    putchar('\n');  
}
```

```
>>>> $ 50-get-put  
x  
x  
>>>> $ 50-get-put  
abcde  
a
```

```
=====
```

```
#include <stdio.h>
```

```
void main (void) {  
    int cc;  
    while((cc = getchar()) != EOF) {  
        putchar(cc);  
    }  
}
```

```
>>>> $ 51-get-put-loop  
xxxx  
xxxx
```

52-open-close

Coming Soon

53-file-pointer

Coming Soon

54-write

Coming Soon

Coming Soon

Coming Soon

Coming Soon

Coming Soon

Coming Soon

60-readwrite

Coming Soon

The End

- This is the end of the presentation.