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

REV098 14-NOV-2017

OS172 | INT TU/TH 13:00-15:00 | EXT TH 19:00-21:50

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: Retreat
Cadangan	30 Nov - 09 Des 2017	
Ming. UAS	10 Des - 23 Des 2017	

Agenda I

Agenda II

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 $\rightarrow$ [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].

Programming

- `putchar(char)`
- `getpid()`
- `getppid()`
- `sprintf(char*, const chat*)`
- `fflush(NULL)`
- `MSIZE1 (10k) MSIZE2 (20k) MSIZE3 (50k) MSIZE4 (100k) MSIZE5 (1M) MSIZE6 (10M) MSIZE1`
- `top`
  - PID (Process Id), PPID (Parent PID), %MEM (Memory), VIRT (Virtual Image KiB), RES (Residen Size KiB), SHR (Shared Memory KiB), SWAP (Swapped Size KiB), CODE (Code Size KiB), DATA (Data+Stack KiB), USED (Res+Swap Size KiB).
  - Save: `~/.toprc`
  - `top -b -n 1 -pYOUR_PID`
- `malloc(size_t)`
- `free(void*)`
- `system(const char*)`

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
```

Linux Libraries

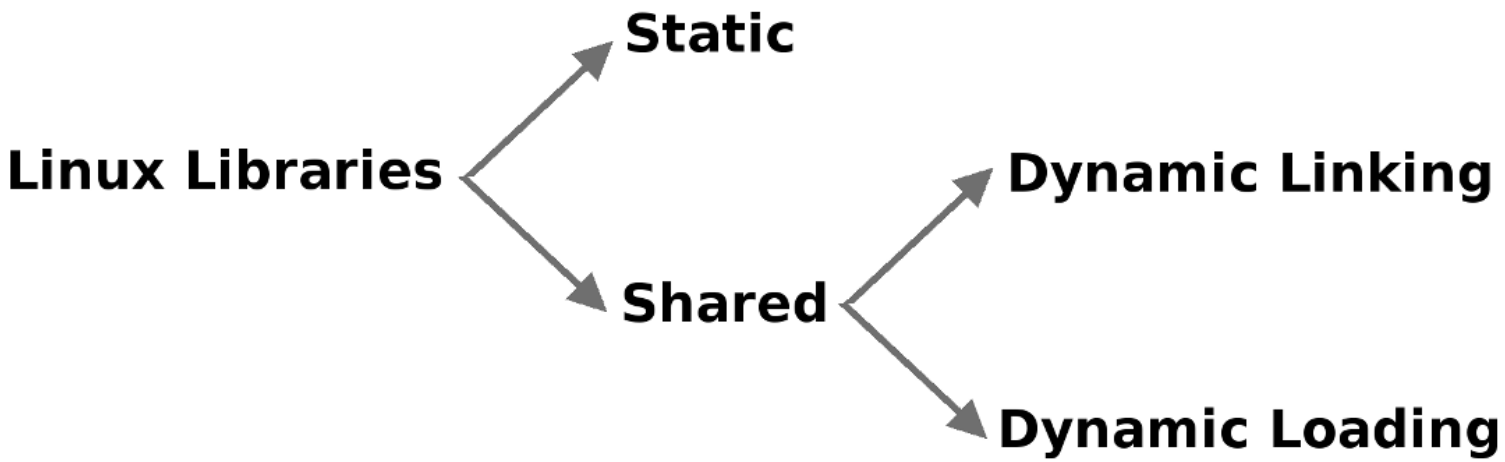

Figure: Linux Libraries

- Static Libraries (embeded 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) {                                    // main-1:    *ptrInt1 = 1
   function1();                                      // function2:     iif2 = 2
   printf("main-1:    *ptrInt1 = %d\n", *ptrInt1);  // main-2:    *ptrInt1 = 1
   function2(*ptrInt1);                              // main-3:     varInt1 = 1
   printf("main-2:    *ptrInt1 = %d\n", *ptrInt1);  // function3:     iif3 = 2
   printf("main-3:     varInt1 = %d\n",  varInt1);  // main-4:     varInt1 = 2
   function3(&varInt1);
   printf("main-4:     varInt1 = %d\n",  varInt1);
}
```

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

```
* === umask() ===
* int open(const char* pathname, int flags, mode_t mode);
* === FLAGS: ===
* O_RDONLY Open the file so that it is read only.
* O_WRONLY Open the file so that it is write only.
* O_RDWR   Open the file so that it can be read from and written to.
* O_APPEND Append new information to the end of the file.
* O_TRUNC  Initially clear all data from the file.
* O_CREAT  If the file does not exist, create it.
             You must include the third parameter.
* O_EXCL   With O_CREAT: exists, the call will fail.
* === MODE ===
* S_IRWXU  00700 user (file  owner) has read, write and execute permission
* S_IRUSR  00400 user has read permission
* S_IWUSR  00200 user has write permission
* S_IXUSR  00100 user has execute permission
*
* S_IRWXG  00070 group has read, write and execute permission
* S_IRGRP  00040 group has read permission
* S_IWGRP  00020 group has write permission
* S_IXGRP  00010 group has execute permission
*
* S_IRWXO  00007 others have read, write and execute permission
* S_IROTH  00004 others have read permission
* S_IWOTH  00002 others have write permission
* S_IXOTH  00001 others have execute permission
```

52-open-close (2)

```
#define  FILE1 "demo-file1.txt"
#define  FILE2 "demo-file2.txt"
#define  FILE3 "demo-file3.txt"

#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
void main(void) {
   char*  file1=FILE1;
   char*  file2=FILE2;
   char*  file3=FILE3;

   int fd; /* to hold a file descriptor */
   /* umask(0);    ******************** */
   fd = open (file1, O_CREAT | O_RDWR, S_IRWXU);
   close(fd);
   fd = open (file2, O_CREAT | O_RDWR, S_IRWXU|S_IRGRP|S_IWGRP|S_IROTH);
   close(fd);
   fd = open (file3, O_CREAT | O_RDWR, 0711);
   close(fd);
   fd = open (file3, O_CREAT | O_RDWR, 0700);
   close(fd);
}

>>>>> $ ls -al demo-file[234].txt
-rwxr--r-- 1 demo demo  0 Oct  5 17:49 demo-file2.txt
-rwx--x--x 1 demo demo  0 Oct  5 17:49 demo-file3.txt
-rw-r--r-- 1 demo demo 75 Oct  5 17:49 demo-file4.txt
>>>>> $
```

53-file-pointer

```
#define  FILE4 "demo-file4.txt"
#include <stdio.h>
#include <stdlib.h>

void main(void) {
   FILE* fp;
   int   cc;

   printf ("*** Open and listing file %s ***\n\n", FILE4);
   if ((fp=fopen(FILE4, "r")) == NULL) {
      printf("fopen error...\n");
      exit(1);
   }
   while((cc=fgetc(fp)) != EOF) {
      printf("%c", cc);
   }
   printf("\n");
   fclose(fp);
}
```

```
*** Open and listing file demo-file4.txt ***

Line 1: Blah Blah Blah 1
Line 2: Blah Blah Blah 2
Line 3: Blah Blah Blah 3
```

54-write

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define  FILE5   "demo-file5.txt"
static char* str1 = "AAAXBBB\n";
static char* str2 = "CCC\n";

void main(void) {
   int fd1, fd2;
   fd1 = open (FILE5, O_RDWR | O_CREAT, 0644);
   fd2 = open (FILE5, O_RDWR | O_CREAT, 0644);
   printf("File Descriptors --- fd1 = %d, fd2 = %d\n", fd1, fd2);
   write(fd1, str1, strlen(str1));
   write(fd2, str2, strlen(str2));
   close(fd1);
   close(fd2);
   printf("See output file %s\n", FILE5);
}


########################################
File Descriptors --- fd1 = 3, fd2 = 4
See output file demo-file5.txt

########################################
demo-file5.txt:
CCC
BBB
```

55-write

```
#define  FILE6  "demo-file6.txt"
char     buf1[] = "abcdefgh";
char     buf2[] = "ABCDEFGH";

void main(void) {
   int fd;
   fd = creat(FILE6, 0644);
   if (fd < 0) {
      perror("creat error");
      exit(1);
   }
   if (write(fd, buf1, 8) != 8) {
      perror("buf1 write error");
      exit(1);
   } /* offset now = 8 */
   if (lseek(fd, 32, SEEK_SET) == -1) {
      perror("lseek error");
      exit(1);
   } /* offset now = 32 */
   if (write(fd, buf2, 8) != 8) {
      perror("buf2 write error");
      exit(1);
   } /* offset now = 40 */
   close(fd);
   printf("Run: hexdump -c %s\n", FILE6);
}
```

```
>>>>> $ hexdump -c demo-file6.txt
0000000   a   b   c   d   e   f   g   h  \0  \0  \0  \0  \0  \0  \0  \0
0000010  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020   A   B   C   D   E   F   G   H
```

56-copy

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define  BUF_SIZE 16

void main(int argc, char* argv[])
{
   int          fdread, fdwrite;
   unsigned int total_bytes = 0;
   ssize_t      nbytes_read, nbytes_write;
   char buf[BUF_SIZE];
   if (argc != 3) {
      printf("Usage: %s source destination\n",
      argv[0]);
      exit(1);
   }
   fdread = open(argv[1], O_RDONLY);
   if (fdread < 0) {
      perror("Failed to open source file");
      exit(1);
   }
   fdwrite = creat(argv[2], S_IRWXU);
   if (fdwrite < 0) {
      perror("Failed to open destination file");
      exit(1);
   }
```

56-copy (2)

```
   do {
      nbytes_read = read(fdread, buf, BUF_SIZE);
      if (nbytes_read < 0) {
         perror("Failed to read from file");
         exit(1);
      }
      nbytes_write = write(fdwrite, buf, nbytes_read);
      if (nbytes_write < 0) {
         perror("Failed to write to file");
         exit(1);
      }
   } while (nbytes_read > 0);
   close(fdread);
   close(fdwrite);
   exit(0);
}

##########################################################

>>>>> $ ./56-copy demo-file4.txt demo-copy.txt
>>>>> $ ls -al demo-file4.txt demo-copy.txt
-rwx------ 1 demo demo 75 Oct  5 18:12 demo-copy.txt
-rw-r--r-- 1 demo demo 75 Oct  5 17:49 demo-file4.txt
>>>>> $
```

57-dup

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#define FILE1 "demo-file7.txt"

static char* str1 = "AAAXBBB\n";
static char* str2 = "CCC\n";

Coming Soon
void
   int fd1, fd2;
   fd1 = open (FILE1, O_RDWR | O_CREAT, 0644);
   fd2 = dup(fd1);
   printf("File Descriptors --- fd1 = %d, fd2 = %d\n", fd1, fd2);
   write(fd1, str1, strlen(str1));
   write(fd2, str2, strlen(str2));
   close(fd1);
   close(fd2);
   printf("**** Please check file %s *****\n", FILE1);
   printf("**** Compare with 54-write\n");
}
##############################################
>>>>> $ 57-dup
File Descriptors --- fd1 = 3, fd2 = 4
**** Please check file demo-file7.txt *****
**** Compare with 54-write
>>>>> $ cat demo-file7.txt
AAAXBBB
CCC
>>>>> $
```

58-dup2

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#define FILE1 "demo-file8.txt"

static char* str1 = "AAAXBBB\n";
static char* str2 = "CCC\n";

void main(void) {
   int fd1, fd2;
   fd1 = open (FILE1, O_RDWR | O_CREAT, 0644);
   dup2(fd1, fd2);
   printf("File Descriptors --- fd1 = %d, fd2 = %d\n", fd1, fd2);
   write(fd1, str1, strlen(str1));
   write(fd2, str2, strlen(str2));
   close(fd1);
   close(fd2);
   printf("**** Please check file %s *****\n", FILE1);
   printf("**** Compare with 54-write\n");
}
##########################################
>>>>> $ 58-dup2
File Descriptors --- fd1 = 3, fd2 = 0
**** Please check file demo-file8.txt *****
**** Compare with 54-write
>>>>> $ cat demo-file8.txt
AAAXBBB
CCC
>>>>> $
```

59-io

```
#include <stdio.h>
#include .....
#define FILE1 "demo-file9.txt"

void main(void) {
   int fd1, fd2;
   char strvar[100];
   printf ("***** Please check file %s ***** *****\n", FILE1);
/* BLOCK ***************
   close(STDERR_FILENO);
   close(STDOUT_FILENO);
   BLOCK *************** */
   fd1 = open (FILE1, O_RDWR | O_CREAT | O_TRUNC, 0644);
   fd2 = dup(fd1);
   printf(         "AAAAA print to standard output!!\n");
   fprintf(stdout, "BBBBB print to standard output!!\n");
   fprintf(stderr, "CCCCC print to standard error!!!\n");
   sprintf(strvar, "DDDDD print to fd1=%d!!!\n", fd1);
   dprintf(fd1,    "%s", strvar);
   dprintf(fd2,    "EEEEE print to fd2=%d!!!\n", fd2);
   close(fd1);
   close(fd2);
}
>>>>> $ 59-io ; echo "^^^^^";cat demo-file9.txt
***** Please check file demo-file9.txt ***** *****
AAAAA print to standard output!!
BBBBB print to standard output!!
CCCCC print to standard error!!!
^^^^^
DDDDD print to fd1=3!!!
EEEEE print to fd2=4!!!
```

59-io (2)

```
#include <stdio.h>
#include .....
#define FILE1 "demo-file9.txt"

void main(void) {
   int fd1, fd2;
   char strvar[100];
   printf ("***** Please check file %s ***** *****\n", FILE1);
   close(STDERR_FILENO);
/* BLOCK ***************
   close(STDOUT_FILENO);
   BLOCK *************** */
   fd1 = open (FILE1, O_RDWR | O_CREAT | O_TRUNC, 0644);
   fd2 = dup(fd1);
   printf(         "AAAAA print to standard output!!\n");
   fprintf(stdout, "BBBBB print to standard output!!\n");
   fprintf(stderr, "CCCCC print to standard error!!!\n");
   sprintf(strvar, "DDDDD print to fd1=%d!!!\n", fd1);
   dprintf(fd1,    "%s", strvar);
   dprintf(fd2,    "EEEEE print to fd2=%d!!!\n", fd2);
   close(fd1);
   close(fd2);
}
>>>>> $ 59-io ; echo "^^^^^";cat demo-file9.txt
***** Please check file demo-file9.txt ***** *****
AAAAA print to standard output!!
BBBBB print to standard output!!
^^^^^
CCCCC print to standard error!!!
DDDDD print to fd1=2!!!
EEEEE print to fd2=3!!!
```

59-io (3)

```
#include <stdio.h>
#include .....
#define FILE1 "demo-file9.txt"

void main(void) {
   int fd1, fd2;
   char strvar[100];
   printf ("***** Please check file %s ***** *****\n", FILE1);
   close(STDERR_FILENO);
   close(STDOUT_FILENO);
/* BLOCK ***************
   BLOCK *************** */
   fd1 = open (FILE1, O_RDWR | O_CREAT | O_TRUNC, 0644);
   fd2 = dup(fd1);
   printf(         "AAAAA print to standard output!!\n");
   fprintf(stdout, "BBBBB print to standard output!!\n");
   fprintf(stderr, "CCCCC print to standard error!!!\n");
   sprintf(strvar, "DDDDD print to fd1=%d!!!\n", fd1);
   dprintf(fd1,    "%s", strvar);
   dprintf(fd2,    "EEEEE print to fd2=%d!!!\n", fd2);
   close(fd1);
   close(fd2);
}
>>>>> $ 59-io ; echo "^^^^^";cat demo-file9.txt
***** Please check file demo-file9.txt ***** *****
^^^^^
AAAAA print to standard output!!
BBBBB print to standard output!!
CCCCC print to standard error!!!
DDDDD print to fd1=1!!!
EEEEE print to fd2=2!!!
```

60-readwrite

```
#define FILE1 "demo-fileA.txt"
#define  OLOOP 10
#define  ILOOP 3650
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <fcntl.h>
#include <dirent.h>
void rwfile (char* fname);
void dirfile(char* dname);
void error  (char* msg);
/* MAIN ============================ */
void main(void) {
   printf("Listing current directory...\n");
   dirfile(".");
   printf("Testing read-write speed...\n");
   rwfile(FILE1);
}
/* DIRFILE ======================== */
void dirfile(char* dname) {
   DIR*            ddir;
   struct dirent*  dp;
   printf("     ");
   ddir = opendir(dname);
   if (ddir != NULL) {
      while ((dp=readdir(ddir))!= NULL)
         printf("%s  ", dp->d_name);
      closedir(ddir); }
   printf("\n\n"); }
```

60-readwrite (2)

```
/* ERROR ========================== */
void error(char* msg){
   perror(msg);
   exit(0); }
/* RWFILE ========================== */
void rwfile(char* fname) {
   time_t tt;
   int    fd, ii, jj;
   char   buf[] = "Achtung... Achtung... AAAA BBBB CCCC DDDD\n";
   time(&tt);
   for (ii=0;ii<OLOOP;ii++) {
      if ((fd=creat(fname,00644)) < 0 )
         error("RWFILE: can not create file\n");
      for (jj=0;jj<ILOOP;jj++) {
         write(fd,buf,sizeof(buf)-1);
         fsync(fd); }
      close(fd);
      putchar('.');
      fflush(NULL); }
   tt=time(NULL)-tt;
   putchar('\n');
   printf("Total time: %d seconds\n", (int) tt);
}
######################
>>>>> $ time 60-readwrite
Listing current directory...
     .shsh  52-open-close.c  demo-file4.txt  02-pointers.c ...
Testing read-write speed...
..........
Total time: 10 seconds
real  0m9.998s  -----  user   0m0.024s  -----  sys 0m0.576s
```

The End

- This is the end of the presentation.