



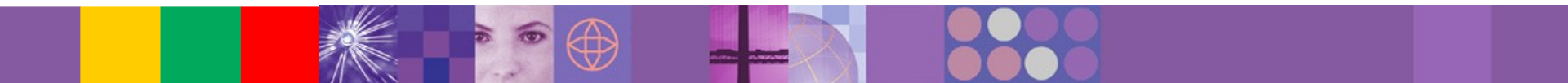
IBM Software Group

# IBM AIX C 语言开发讲座 – 概述

WebSphere. software

刘睿

liurui@cn.ibm.com



 e-business software

# 概要

1. AIX 编译器以及线程方式
2. AIX 内存管理
3. AIX 的函数库

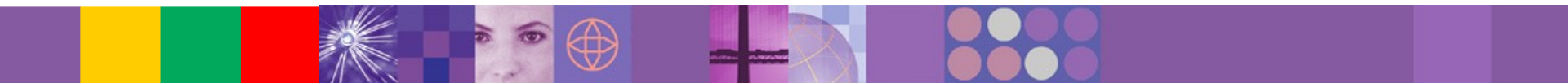




IBM Software Group

# 1. AIX 编译器以及线程方式

WebSphere. software



 e-business software

# 编译器命令何其多！

- vac 不同形式的命令的配置：/etc/vac.cfg
- 各种类型的编译命令：

```
CC      CC_R      CC_R4      CC_R7
C89     C89_R     C89_R4    C89_R7
C99     C99_R     C99_R4    C99_R7
XLC     XLC_R     XLC_R4    XLC_R7
```

// 支持 128 位数据

```
CC128   CC128_R   CC128_R4  CC128_R7
C89_128 C89_128_R C89_128_R4 C89_128_R7
C99_128 C99_128_R C99_128_R4 C99_128_R7
XLC128  XLC128_R  XLC128_R4 XLC128_R7
```



# 编译器命令何其多！（续）

- 编译器分类举例

  - xlc – 单线程

  - xlc\_r - POSIX threads

  - xlc\_r4 - Version 4 draft of the POSIX threads standard (DCE)

  - xlc\_r7 - Version 7 draft of the POSIX threads standard

  - Note: /usr/include/pthread.h 和 /usr/lib/libpthread.a

- 编译器的最常用的命令行选项

  - c , -D , -g , -I , -L , -o , -q64( 支持 64 位 , 默认 -q32) , -qcpluscmt( 支持 C++ 注释 )



## 线程要点

- POSIX threads (Pthreads) Standard - IEEE POSIX 1003.1c standard.2  
“/usr/include/pthread.h” and “/usr/lib/libpthreads.a”.
- 老版本 AIX 的限制
- 非可重入函数，例如：  
**char \*asctime (const struct tm \*Tm);**  
**char \*ctime (const time\_t \*Clock);**



## libc 中的非可重入子例程

asctime getgrent gsignal setkey auditread getgrgid hcreate setlogmask closelog  
getgrnam hdestroy setnetent crypt getgroupsbyuser hsearch setnetgrent ctime getgrset  
inet\_ntoa setprotoent dirname gethostbyaddr initstate setpwent drand48  
gethostbyname inetgr setpwnfile ecvt gethostent iso\_addr setrpcent endttyent getlogin  
iso\_ntoa setservent encrypt getnetbyaddr jrand48 setstate asctime getnetbyname l64a  
setttyent endfsent getnetent lcong48 setutent endfsent getnetgrent link\_ntoa setutxent  
endgrent getopt localtime srand48 endhostent getprotobyname lrand48 srandom  
endnetent getprotobynumber mrand48 ssignal endnetgrent getprotoent mtime strerror  
endprotoent getpwent ndutent strtok endpwent getpwnam nrand48 syslog endrpcent  
getpwuid ns\_ntoa ttyname endservent getrpcbyname openlog utmpname endttyent  
getrpcbynumber pututline wcstok endutxent getrpercent pututxline erand48  
getservbyname rand ether\_aton getservbyport random ether\_ntoa getservent rcmd  
fcvt fgetgrent getttyent rcmd2 fgetpwent gettynam readdir getdate getuinfo rexec  
getfsent getutent re\_comp getfsent getutid re\_exec getfsfile getutline seed48 getfsfile  
getutxent setfsent getfsspec getutxid setgrent getfstype getutxline sethostent

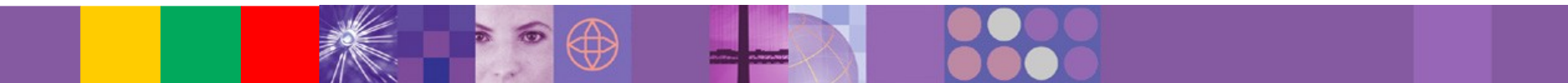




IBM Software Group

## 2. AIX 内存管理

WebSphere. software



## 系统调用 free 并不释放内存页！

- C free 与 C++ delete 做什么？
- 为什么长期运行的进程，比如 cicsas 进程可能造成整机的内存紧张。
- 释放进程不再使用的内存页。

`mallopt(M_DISCLAIM, 0);`

其它办法包括使用 `disclaim` 函数

或者使用环境变量：

“`MALLOCOPTIONS=disclaim`” 或者  
“`MALLOCDISCLAIM=true`”。



# 系统调用 free 并不释放内存页！（续）

```
#define BUF_SIZE 200000000
```

```
int main(){  
    char *p;
```

```
    p= malloc( BUF_SIZE );  
    if(p){  
        puts( "Malloc OK!" );  
        memset( p, 6, BUF_SIZE );  
        free(p);  
        mallopt(M_DISCLAIM, 0);  
    }  
    else{  
        puts( "Malloc failed!" );  
    }  
}
```

```
while(1){  
    p= malloc(100);  
    free(p);
```

```
    sleep(1);  
}  
return 0;  
}
```



## 想分配大块的数据怎么办？

- ulimit 的双重限制
  - 系统的默认设置记录在 /etc/security/limits 文件 (单位 512byte)。
  - ulimit 命令  
例如 (默认是 131072 Kbyte) : ulimit -d unlimited
- 对 32 位的程序，如果数据大于一个段 (256M)，需要设置 MAXDATA 参数
  - 设置环境变量  
export LDR\_CNTRL=MAXDATA=0x20000000  
./testmalloc  
unset LDR\_CNTRL  
或者：  
LDR\_CNTRL=MAXDATA=0x20000000 ./testmalloc 300000000
  - 使用编译参数 bmaxdata  
cc -o testmalloc -bmaxdata:0x20000000 testmalloc.c



## 想分配大块的数据怎么办？（续）

- LDR\_CNTRL environment variable  
export LDR\_CNTRL=MAXDATA=0x20000000  
start\_process  
unset LDR\_CNTRL
- LDR\_CNTRL Settings LDP\_CNTRL Setting Total Number of Segments  
Process Memory Limit

Unset	0(default)	256 MB
LDR_CNTRL=MAXDATA=0x10000000	1	256 MB
LDR_CNTRL=MAXDATA=0x20000000	2	512 MB
LDR_CNTRL=MAXDATA=0x30000000	3	768 MB
LDR_CNTRL=MAXDATA=0x40000000	4	1 GB
LDR_CNTRL=MAXDATA=0x50000000	5	1.24 GB
LDR_CNTRL=MAXDATA=0x60000000	6	1.5 GB
LDR_CNTRL=MAXDATA=0x70000000	7	1.75 GB
LDR_CNTRL=MAXDATA=0x80000000	8	2 GB

- If an invalid setting is used for the LDR\_CNTRL environment variable, it will be ignored and the default one-segment usage will be defined.



## AIX 32 位进程的默认内存模式的分段结构

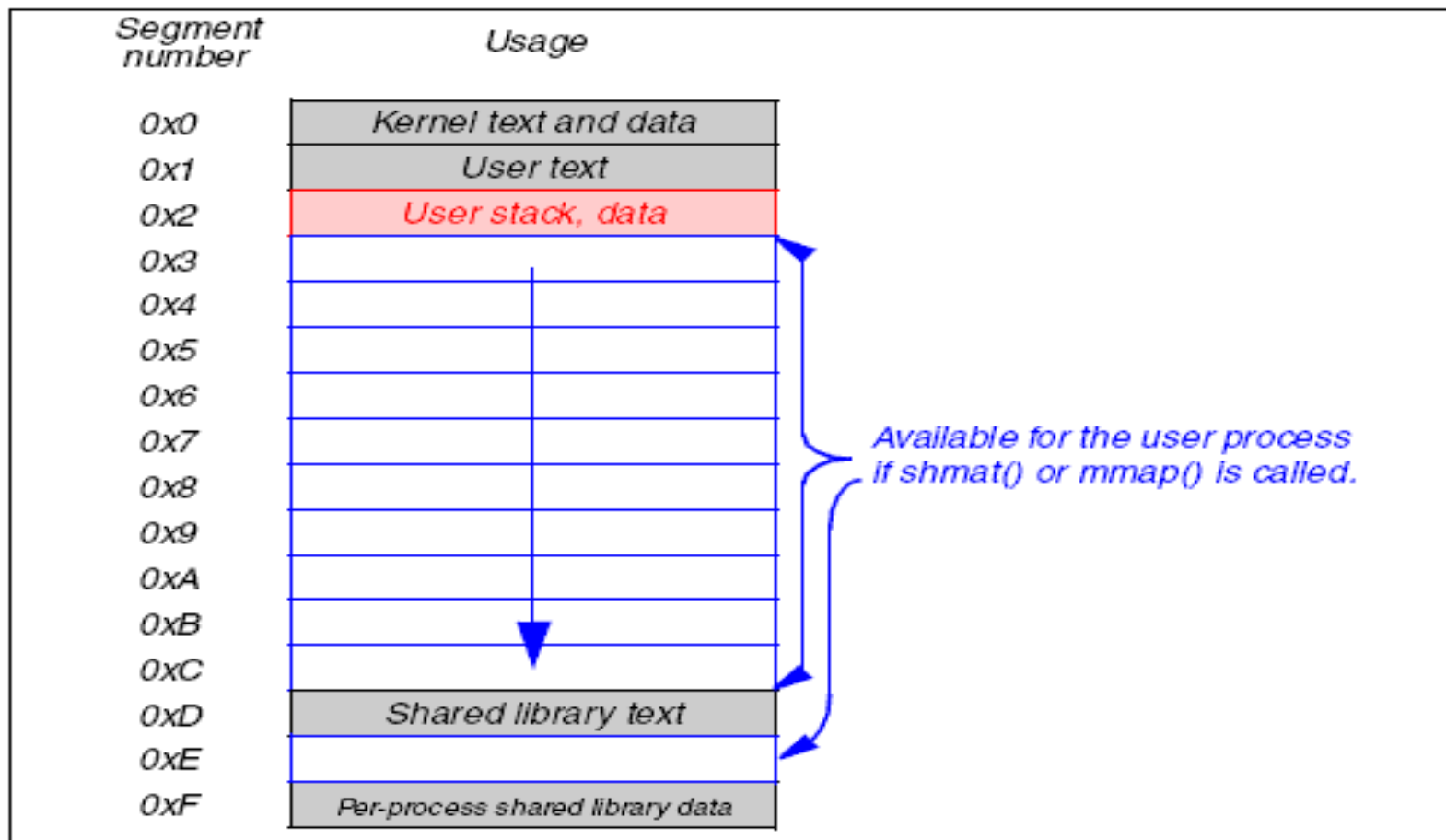


Figure 3-2 Default memory model (segment usage)<sup>4</sup>



## AIX 32 位进程的大内存模式的分段结构

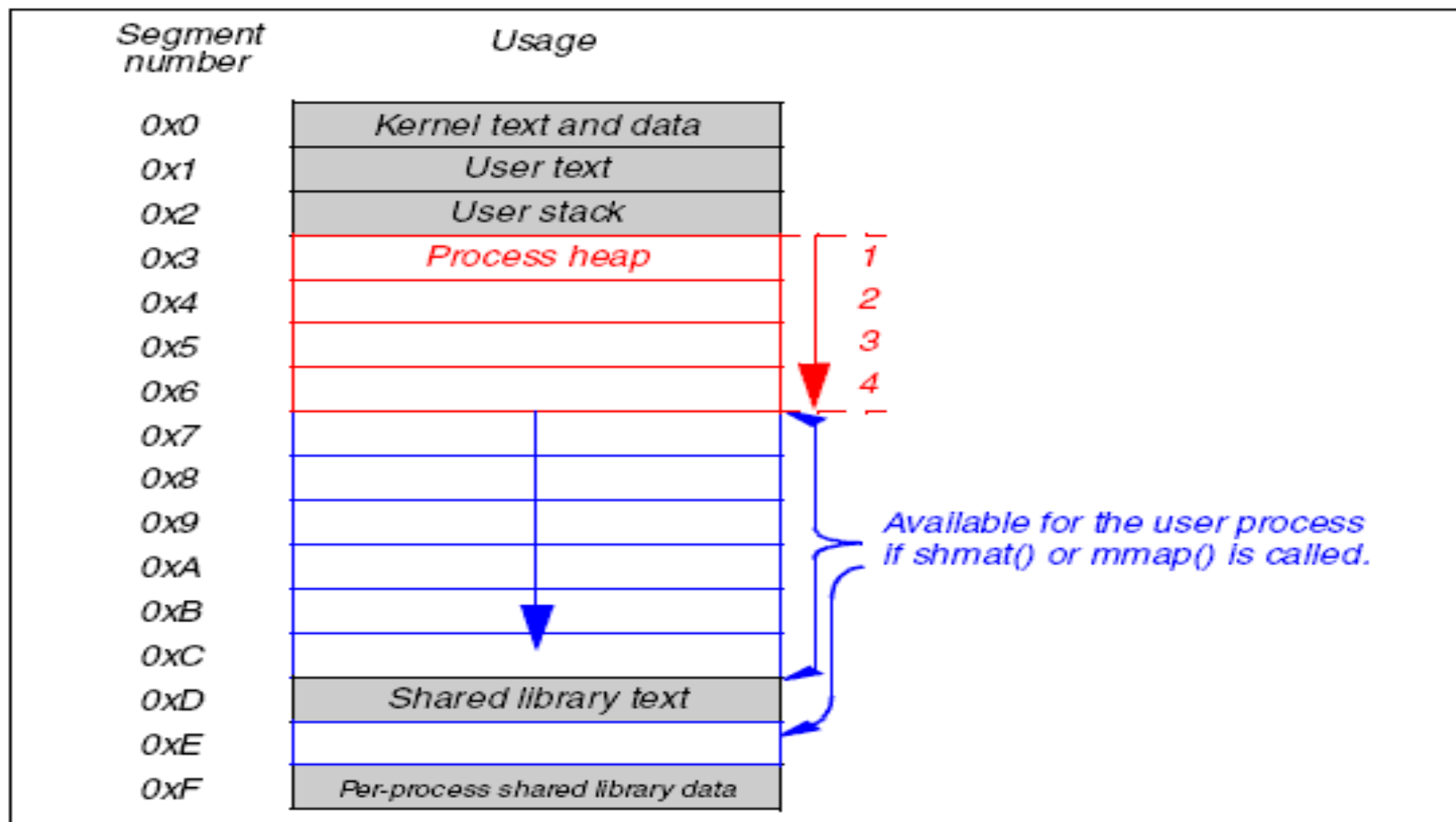


Figure 3-4 Large memory model (segment usage)



## 32 位进程的共享库代码怎样被共享

- 共享库的代码都放在 => **系统的 D 段！**  
好处：性能高，省内存。
- 多系统的机器怎样充分利用 D 段？  
使用**命名共享库的内存段！**  
例如：LDR\_CNTRL=NAMEDSHLIB=CICS
- 何谓“private shared objects”？



## 32 位进程分配共享内存时 (shmat) 遇到的麻烦

- 在默认情况下的段对齐 => 最多分配 11 块！
- 设置 EXTSHM=ON ，注意兼容性！



## 与共享库的内存使用相关的常用命令

- 观察系统加载的共享库所占的地址空间：  
genkld
- 观察当前运行的进程加载的共享库所占的地址空间：  
genld -l
- 观察进程的内存使用：  
svmon -P <pid>
- 观察进程的共享模块内存使用：  
procmap <pid>
- 清除共享内存段：  
slibclean



## 命令 genklid 的输出举例

TEXT	ADDRESS	SIZE	FILE
	D22BE100	36732	/USR/LIB/LIBPTOOLS.A[SHR.O]
	D22F5080	297DE	/USR/LIB/LIBTRACE.A[SHR.O]
	D22938A0	2A34E	/USR/LIB/LIBSM.A[SHR.O]
	D21CD100	44BD0	/USR/LIB/LIBDIAG.A[SHR.O]
	D2212100	1A8F1	/USR/LIB/LIBASL.A[SHR.O]
	D222D100	17A4D	/USR/LIB/LIBCUR.A[SHR.O]
	D2278FC0	1A011	/USR/LIB/LIBCUR.A[SHR32C.O]
	D0873100	36B9	/USR/LIB/LIBERRLOG.A[SHR.O]
	D1420100	6801	/USR/LIB/LIBSSA.A[SHR.O]
	D217C000	1FB0F	/USR/LPP/CICS/LIB/LIBCICSENC.SA.O
	D219C108	306DE	/USR/LPP/CICS/LIB/LIBENCCLIENT.A[ENCCLIENT_SHR.O]
	D2008100	1735E9	/USR/LIB/LIBDBX.A[SHR.O]
	D088E100	5B2B	/USR/LIB/LIBCDEBUG.A[SHR.O]
	D0B8E200	14DD2	/USR/LIB/LIBDBX.A[COREDEBUG.O]
	...	...	...



# 命令 genId -l 的输出举例

```
... ..
PROC_PID: 344110  PROC_NAME: AIOSERVER

PROC_PID: 348258  PROC_NAME: CICSAS
10000000  608B  CICSAS
D0343100  2514  /HOME/DB2INST1/SQLLIB/SECURITY32/PLUGIN/IBM/CLIENT/IBMOSAETHCLIENT.A[SHR.O]
D48E4000  15723F /HOME/DB2INST1/SQLLIB/LIB32/ICC/OSSLIB/LIBCRYPTO.SO.0.9.7
D03250F8  1E0   /USR/LIB/LIBDL.A[SHR.O]
D13B7000  1D160 /HOME/DB2INST1/SQLLIB/LIB32/ICC/ICCLIB/LIBICCLIB.SO
D24F8100  12D96F /USR/LPP/CICS/LIB/LIBENCDFS.A[ENCDFS_SHR.O]
D09B6100  32830 /USR/LPP/CICS/LIB/LIBDAMFSRT.A[LIBDAMFS_SHR.O]
30351000  F36   /USR/LPP/CICS/BIN/CICSSWSFS
D139A100  1C110 /OPT/IBM/DB2/V9.5/LIB32/LIBDB2DASCMN.A[SHR.O]
D1386100  13953 /OPT/IBM/DB2/V9.5/LIB32/LIBDB2LOCALE.A[SHR.O]
D09E9100  8261  /OPT/IBM/DB2/V9.5/LIB32/LIBDB2TRCAPI.A[SHR.O]
D0904100  4B20  /OPT/IBM/DB2/V9.5/LIB32/LIBDB2INSTALL.A[SHR.O]
D4293100  650064 /OPT/IBM/DB2/V9.5/LIB32/LIBDB2G11N.A[SHR.O]
... ..
```



# 命令 svmon -P 的输出举例

```
# SVMON -P 348258
```

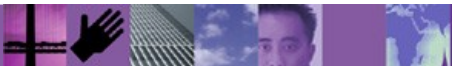
```
-----
PID COMMAND      INUSE  PIN  PGSP VIRTUAL 64-BIT MTHRD 16MB
348258 CICSAS      77852 65556 0 76315  N  Y  N

VSID  ESID TYPE DESCRIPTION          PSIZE INUSE  PIN PGSP VIRTUAL
0     0     0     WORK KERNEL SEGMENT (LPGG_VSID=0) L  16  16  0  0  16
1D309D D     0     WORK SHARED LIBRARY TEXT      S  7917  0  0  0  7917
F1F8E  C     0     WORK SHARED MEMORY SEGMENT   S  1537  0  0  0  1537
1413B5 -     0     CLNT /DEV/HD9VAR:103082      S  1410  0  -  -  -
60787  F     0     WORK SHARED LIBRARY DATA    S  544  0  0  0  544
1F075E 3     0     WORK WORKING STORAGE         S  432  0  0  0  432
141235 A     0     WORK SHARED MEMORY SEGMENT   S  196  0  0  0  196
71FC6  -     0     CLNT /DEV/HD2:134562        S  73  0  -  -  -
91108  7     0     WORK SHARED MEMORY SEGMENT   S  58  0  0  0  58
E07AF  -     0     WORK                          S  53  17  0  53  -
10760  2     0     WORK PROCESS PRIVATE         S  38  3  0  0  38
E13CF  -     0     CLNT /DEV/HD9VAR:103098      S  35  0  -  -  -
713A6  1     0     CLNT CODE,/DEV/HD2:133928    S  13  0  -  -  -
141075 -     0     CLNT /DEV/HD9VAR:102501      S  3  0  -  -  -
150074 -     0     CLNT /DEV/HD2:20374          S  2  0  -  -  -
211C3  8     0     WORK SHARED MEMORY SEGMENT   S  2  0  0  0  2
31062  B     0     WORK SHARED MEMORY SEGMENT   S  2  0  0  0  2
101FD1 -     0     CLNT /DEV/HD9VAR:102449      S  1  0  -  -  -
611C7  6     0     WORK WORKING STORAGE         S  0  0  0  0  0
151034 4     0     WORK WORKING STORAGE         S  0  0  0  0  0
20743  5     0     WORK WORKING STORAGE         S  0  0  0  0  0
1B1FBA -     0     CLNT /DEV/HD9VAR:99729      S  0  0  -  -  -
```



# 命令 procmap 的输出举例

```
# PROCMAP 348258
348258 : CICSAS CICS01 7340042 101 A0000000 B0000000
10000000      24K READ/EXEC      CICSAS
30000398      1K READ/WRITE      CICSAS
D0343100      9K READ/EXEC      /HOME/DB2INST1/SQLLIB/SECURITY32/PLUGIN/IBM/CLIENT/IBMOSAUTHCLIENT.A[SHR.O]
F12B3CE5      1K READ/WRITE      /HOME/DB2INST1/SQLLIB/SECURITY32/PLUGIN/IBM/CLIENT/IBMOSAUTHCLIENT.A[SHR.O]
D48E4000     1372K READ/EXEC      /HOME/DB2INST1/SQLLIB/LIB32/ICC/OSSLIB/LIBCRIPTO.SO.0.9.7
F1278988     231K READ/WRITE      /HOME/DB2INST1/SQLLIB/LIB32/ICC/OSSLIB/LIBCRIPTO.SO.0.9.7
D03250F8      0K READ/EXEC      /USR/LIB/LIBDL.A[SHR.O]
F03BE0F8      0K READ/WRITE      /USR/LIB/LIBDL.A[SHR.O]
D13B7000     116K READ/EXEC      /HOME/DB2INST1/SQLLIB/LIB32/ICC/ICCLIB/LIBICCLIB.SO
F1273AB8      15K READ/WRITE      /HOME/DB2INST1/SQLLIB/LIB32/ICC/ICCLIB/LIBICCLIB.SO
D24F8100     1206K READ/EXEC      /USR/LPP/CICS/LIB/LIBENCsFs.A[ENCsFs_SHR.O]
F060CF00      74K READ/WRITE      /USR/LPP/CICS/LIB/LIBENCsFs.A[ENCsFs_SHR.O]
D09B6100     202K READ/EXEC      /USR/LPP/CICS/LIB/LIBDAMFSRT.A[LIBDAMFS_SHR.O]
F0620D00      11K READ/WRITE      /USR/LPP/CICS/LIB/LIBDAMFSRT.A[LIBDAMFS_SHR.O]
30351000      3K READ/EXEC      /USR/LPP/CICS/BIN/CICSSWSFS
30352150      0K READ/WRITE      /USR/LPP/CICS/BIN/CICSSWSFS
D139A100     112K READ/EXEC      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2DASCMN.A[SHR.O]
F062DB04      8K READ/WRITE      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2DASCMN.A[SHR.O]
D1386100     78K READ/EXEC      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2LOCALE.A[SHR.O]
F076C720     19K READ/WRITE      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2LOCALE.A[SHR.O]
D09E9100     32K READ/EXEC      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2TRCAPI.A[SHR.O]
F0348088     2K READ/WRITE      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2TRCAPI.A[SHR.O]
D0904100     18K READ/EXEC      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2INSTALL.A[SHR.O]
F062982E      3K READ/WRITE      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2INSTALL.A[SHR.O]
D4293100     6464K READ/EXEC      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2G11N.A[SHR.O]
F0DE7009      89K READ/WRITE      /OPT/IBM/DB2/V9.5/LIB32/LIBDB2G11N.A[SHR.O]
... ..
```





IBM Software Group

# 3. AIX 的函数库

WebSphere. software



 e-business software

## 千万不要被文件名称后缀搞晕！— AIX 不一样

- 搞清楚这些后缀：
  - .o 文件？
  - .a 文件？
  - .so 文件？
- 搞清楚这些概念
  - Static object
  - Shared object
  - Library



## 一般的共享库编译 / 链接方式

- 编译 / 链接共享库 ( 注意 -bM:SRE 选项 )

```
xlc -c subs1.c
```

```
xlc -c subs2.c
```

```
ld -o shrsub.so subs1.o subs2.o -bM:SRE -bE:shrsub.exp -bnoentry -lc
```

注意这里特意把后缀写成 .so ，实际上 AIX 上普遍使用的后缀是 .o 。

- ( 可选 ) 创建存档库

```
ar ruv libsub.a shrsub.so
```

- 附录 : shrsub.exp

```
! /opt/lr/shlib/shrsub.so
```

```
*
```

\* Above, we have the full pathname to the shared library object file

\* Below, we have the list of symbols to be exported:

```
*
```

```
func1
```

```
func2
```



## 静态库？共享库？

- 注意 Flags 中的 **SHROBJ**

```
# DUMP -OV /OPT/IBM/DB2/V9.5/LIB32/LIBDB2.A
```

```
/OPT/IBM/DB2/V9.5/LIB32/LIBDB2.A[SHR.O]:
```

```
***OBJECT MODULE HEADER***
```

```
# SECTIONS      SYMBOL PTR      # SYMBOLS      OPT HDR LEN      FLAGS
      7      0X012FABAE      51003          72      0X3002
```

```
FLAGS=( EXEC DYNLOAD SHROBJ DEP_SYSTEM )
```

```
TIMESTAMP = "MAR 28 17:00:39 2008"
```

```
MAGIC = 0X1DF (32-BIT XCOFF)
```

```
***OPTIONAL HEADER***
```

```
TSIZE      DSIZE      BSIZE      TSTART      DSTART
0X00CAA1EC 0X001F7F18 0X0027A0B4 0X10000178 0X20000364
```

```
... ..
```



## AIX 共享库一些特殊用法

- Private shared objects
- Lazy loading
  - blazy linker option
- Dynamic Loading
  - dlopen(), dlsym(), dlclose()...



## 使用共享库编程序

- 以下方式都能编译成功：

```
xlc -o main main.o -L/opt/lr/shlib -lsub
```

```
xlc -o main main.o /opt/lr/shlib/shrsub.so
```

```
xlc -o main main.o /opt/lr/shlib/libsub.so
```

- 共享库也可以支持静态联编，例如：

```
xlc -o main main.o -bstatic -lx -Lnewpath -bdynamic
```

注意这种做法在版本升级后可能导致的兼容性问题



## 共享库的符号解析和库定位规则

- 联编时定位符号所属的库：
  - 默认的库路经 (/usr/lib:/lib)
  - -L 参数的作用
- 执行时如何定位共享库？
  - 默认的库路经 (/usr/lib:/lib)
  - LIBPATH 环境变量



## 共享库的符号解析和库定位规则举例（一）

```
# XLC -O MAIN MAIN.O /OPT/LR/SHLIB/SHRSUB.SO
# DUMP -HV MAIN
```

MAIN:

\*\*\*LOADER SECTION\*\*\*

LOADER HEADER INFORMATION

VERSION#	#SYMTABLEENT	#RELOCENT	LENIDSTR
0x00000001	0x00000009	0x00000012	0x00000044

#IMPFIID	OFFIDSTR	LENSTRTBL	OFFSTRTBL
0x00000003	0x000001d0	0x0000002a	0x00000214

\*\*\*IMPORT FILE STRINGS\*\*\*

INDEX	PATH	BASE	MEMBER
0	/USR/VAC/LIB:/USR/LIB:/LIB		
1		LIBC.A	SHR.O
2	/OPT/LR/SHLIB	SHRSUB.SO	



## 共享库的符号解析和库定位规则举例 (二)

```
# XLC -O MAIN MAIN.O -L/OPT/LR/SHLIB -LSUB  
# DUMP -HV MAIN
```

MAIN:

\*\*\*LOADER SECTION\*\*\*

LOADER HEADER INFORMATION

VERSION#	#SYMTABLEENT	#RELOCENT	LENIDSTR
0x00000001	0x00000009	0x00000012	0x0000004D

#IMPFILID	OFFIDSTR	LENSTRTBL	OFFSTRTBL
0x00000003	0x000001D0	0x0000002A	0x0000021D

\*\*\*IMPORT FILE STRINGS\*\*\*

INDEX	PATH	BASE	MEMBER
0	/OPT/LR/SHLIB:/USR/VAC/LIB:/USR/LIB:/LIB		
1		LIBC.A	SHR.O
2		LIBSUB.A	SHRSUB.SO



## 令人头痛的 AIX Runtime Linking (RTL) 联编方式

- 开发方法的抉择 - 一个哲学问题
- 什么是 RTL (Runtime Linking) ?

Run-time linking enables a program to resolve its referenced symbols at the program load-time rather than link-time (However, in some other UNIX operating systems, resolution of function symbols is deferred until the function is first called). It is the ability to resolve undefined and non-deferred symbols in shared modules after the program execution has already began.



## 链接支持 RTL 的共享对象的标准方式

- `ld -o shsub.so subs1.o subs2.o -G -bE:shsub.exp -bnoentry`  
或：  
`ld -o main.so main.o -G -bexpall -bnoentry`
- `-G = -bM:SRE -brtl -bnortllib -berok -bnosymbolic -bnoautoexp`
  - brtl Enables run-time linking.
  - bnortllib Removes the reference to the run-time linker library.
  - berok Produces the object file even if there are unresolved references.
  - bnosymbolic Assigns this attribute to most symbols exported without an explicit attribute.
  - bnoautoexp Prevents automatic exportation of any symbol.
  - bexpall exports all of its symbols for other shared objects to use
  - bnoentry does not contain any entry points
- ( 可选 ) 创建存档库  
`ar ruv libsub.a shsub.so`



## 使用支持 RTL 的共享对象 ( 库 )

- 联编应用程序 ( 注意 -brtl 是必须的 )

```
xlc -o main -brtl main.so /opt/lr/shlib/shrsub.so
```

或 :

```
xlc -o main -brtl main.so -L/opt/lr/shlib -lsub
```

注意：在使用 -brtl 时，libxxx.so 这种“库模式”也能被 AIX 编译器识别。

- 支持 RTL 的模块的符号如果重绑定：
  - 关键是联编命令行的库的顺序
  - LIBPATH 的作用



# 观察一个支持 RTL 的共享对象的符号表

```
# DUMP -TV MAIN.SO
```

```
MAIN.SO:
```

```
***LOADER SECTION***
```

```
***LOADER SYMBOL TABLE INFORMATION***
```

[INDEX]	VALUE	SCN	IMEX	SCLASS	TYPE	IMPID	NAME
[0]	0x00000000	.DATA	EXP	DS	SECDEF	[NOIMID]	FUNCO
[1]	0x0000000c	.DATA	EXP	DS	SECDEF	[NOIMID]	MAIN
[2]	0x00000000	UNDEF	IMP	DS	EXTREF	..	PUTS
[3]	0x00000000	UNDEF	IMP	DS	EXTREF	..	FUNC1
[4]	0x00000000	UNDEF	IMP	DS	EXTREF	..	FUNC2



# 使用 dump 命令研究程序的设置

```
# DUMP -HOTV MAIN.SO
```

```
MAIN.SO:
```

```
***OBJECT MODULE HEADER***
```

```
# SECTIONS      SYMBOL PTR      # SYMBOLS      OPT HDR LEN      FLAGS
   4      0x00000502          62          72      0x3002
```

```
FLAGS=( EXEC DYNLOAD SHROBJ DEP_SYSTEM )
```

```
TIMESTAMP = "OCT 26 13:43:37 2008"
```

```
MAGIC = 0x1DF (32-BIT XCOFF)
```

```
***OPTIONAL HEADER***
```

```
TSIZE      DSIZE      BSIZE      TSTART      DSTART
0x0000018c 0x00000030 0x00000000 0x00000000 0x00000000
```

```
SNLOADER      SNENTRY      SNTEXT      SNTOC      SNDATA
0x0004      0x0000      0x0001      0x0002      0x0002
```

```
TXTALIGN      DATAALIGN      TOC      VSTAMP      ENTRY
0x0003      0x0002      0x00000018 0x0001      0xffffffff
```

```
MAXSTACK      MAXDATA      SNBSS      MAGIC      MODTYPE
0x00000000 0x00000000 0x0003      0x010B      RE
```



# 使用 dump 命令研究程序的设置 ( 续 )

\*\*\*LOADER SECTION\*\*\*

## LOADER HEADER INFORMATION

VERSION#	#SYMTABLEENT	#RELOCENT	LENIDSTR
0x00000001	0x00000005	0x0000000A	0x0000003B
#IMPFILID	OFFIDSTR	LENSTRTBL	OFFSTRTBL
0x00000002	0x00000110	0x00000000	0x00000000

\*\*\*IMPORT FILE STRINGS\*\*\*

INDEX	PATH	BASE	MEMBER
0	/USR/LPP/CICS/LIB:/USR/LPP/CICSSM/LIB:/OPT/LR/SHLIB		
1	..		

\*\*\*LOADER SYMBOL TABLE INFORMATION\*\*\*

[INDEX]	VALUE	SCN	IMEX	SCLASS	TYPE	IMPID	NAME
[0]	0x00000000	.DATA	EXP	DS	SECDEF	[NOIMID]	FUNC0
[1]	0x0000000C	.DATA	EXP	DS	SECDEF	[NOIMID]	MAIN
[2]	0x00000000	UNDEF	IMP	DS	EXTREF	..	PUTS
[3]	0x00000000	UNDEF	IMP	DS	EXTREF	..	FUNC1
[4]	0x00000000	UNDEF	IMP	DS	EXTREF	..	FUNC2



## 一些有关共享库的命令

- 查询依赖的共享库
  - ldd <file>
  - dump -Hv <file>
  - 使用 dbx 的子命令 map
- 判断目标文件是 64 位还是 32 位
  - file <file>
  - dump -X32\_64 -ov <file>
- 清除装入内存的共享库：slibclean
- 转换共享库为支持 RTL：rtl\_enable <file>
- 得到符号
  - nm < 目标文件 >
  - dump -Tv < 目标文件 >
  - dump -tv a.o | c++filt



**Thank You!**

**谢谢各位！**

