## 实验六：存储管理

## 实验目的：

1. 观察系统存储器使用情况
2. 观察进程使用存储器的情况
3. 掌握通过内存映像文件提高性能的方法
4. 掌握动态内存分配技术

### 预备知识：

1. 存储相关的命令
   1. free 显示系统使用和未被使用的内存数量（可以实时执行）

输出包含的标题有 3 行信息：

Mem。此行包含了有关物理内存的信息。包括以下详细内容：

total。该项显示可用的物理内存总量，单位为 KB。该数字小于安装的物理内存的容量，是因为内核本身也要使用一小部分的内存。

used。该项显示了用于应用程序超速缓存数据的内存容量。

free。该项显示了此时未使用且有效的内存容量。

Shared/buffers/cached。这些列显示了有关内存如何使用的更为详细的信息。

-/+ buffers/cache。Linux 系统中的部分内存用来为应用程序或设备高速缓存数据。这部分内存在需要用于其他目的时可以释放。

free 列显示了调整的缓冲区行，显示释放缓冲区或高速缓存时可以使用的内存容量。

Swap。该行显示有关交换内存利用率的信息。该信息包含全部、已使用和释放的可用内存容量。

* 1. vmstat 报告进程、内存、分页、IO等多类信息（使用手册页）
  2. size 列出目标文件段大小和总大小（使用手册页）

1. /proc文件系统（使用手册页man 5 proc）
   1. /proc/meminfo 内存状态信息
   2. /proc/stat 包含内存页、内存对换等信息。
   3. /proc/$pid/stat 某个进程的信息(包含内存使用信息)
   4. /proc/$pid/maps某个进程的内存映射区信息，包括地址范围、权限、偏移量以及主次设备号和映射文件的索引节点。
   5. /proc/$pid/statm 某个进程的内存使用信息，包括内存总大小、驻留集大小、共享页面数、文本页面数、堆栈页面数和脏页面数。
2. 内存映像文件

内存映像文件是指把一个磁盘文件映像到内存中，二者存在逐字节的对应关系。这样做可以加速I/O操作，并可以共享数据。

3.1 mmap（建立内存映射）

表头文件 #include <unistd.h>

#include <sys/mman.h>

定义函数 void \*mmap(void \*start,size\_t length,int prot,int flags,int fd,off\_t offsize);

函数说明 mmap()用来将某个文件内容映射到内存中，对该内存区域的存取即是直接对该文件内容的读写。参数start指向欲对应的内存起始地址，通常设为NULL，代表让系统自动选定地址，对应成功后该地址会返回。参数length代表将文件中多大的部分对应到内存。

参数 prot代表映射区域的保护方式有下列组合

PROT\_EXEC 映射区域可被执行

PROT\_READ 映射区域可被读取

PROT\_WRITE 映射区域可被写入

PROT\_NONE 映射区域不能存取

参数 flags会影响映射区域的各种特性

MAP\_FIXED 如果参数start所指的地址无法成功建立映射时，则放弃映射，不对地址做修正。通常不鼓励用此旗标。

MAP\_SHARED对映射区域的写入数据会复制回文件内，而且允许其他映射该文件的进程共享。

MAP\_PRIVATE 对映射区域的写入操作会产生一个映射文件的复制，即私人的“写入时复制”（copy on write）对此区域作的任何修改都不会写回原来的文件内容。

MAP\_ANONYMOUS建立匿名映射。此时会忽略参数fd，不涉及文件，而且映射区域无法和其他进程共享。

MAP\_DENYWRITE只允许对映射区域的写入操作，其他对文件直接写入的操作将会被拒绝。

MAP\_LOCKED 将映射区域锁定住，这表示该区域不会被置换（swap）。

在调用mmap()时必须要指定MAP\_SHARED 或MAP\_PRIVATE。参数fd为open()返回的文件描述词，代表欲映射到内存的文件。参数offset为文件映射的偏移量，通常设置为0，代表从文件最前方开始对应，offset必须是分页大小的整数倍。

返回值 若映射成功则返回映射区的内存起始地址，否则返回MAP\_FAILED(－1)，错误原因存于errno 中。

错误代码 EBADF 参数fd 不是有效的文件描述词

EACCES 存取权限有误。如果是MAP\_PRIVATE 情况下文件必须可读，使用MAP\_SHARED则要有PROT\_WRITE以及该文件要能写入。

EINVAL 参数start、length 或offset有一个不合法。

EAGAIN 文件被锁住，或是有太多内存被锁住。

ENOMEM 内存不足。

3.2 munmap（解除内存映射）

表头文件 #include<unistd.h>

#include<sys/mman.h>

定义函数 int munmap(void \*start,size\_t length);

函数说明 munmap()用来取消参数start所指的映射内存起始地址，参数length则是欲取消的内存大小。当进程结束或利用exec相关函数来执行其他程序时，映射内存会自动解除，但关闭对应的文件描述词时不会解除映射。

返回值 如果解除映射成功则返回0，否则返回－1，错误原因存于errno中错误代码EINVAL

参数 start或length 不合法。

1. 动态内存分配

4.1 malloc（配置内存空间）

表头文件 #include<stdlib.h>

定义函数 void \* malloc(size\_t size);

函数说明 malloc()用来配置内存空间，其大小由指定的size决定。

返回值 若配置成功则返回一指针，失败则返回NULL。

4.2 free（释放原先配置的内存）

表头文件 #include<stdlib.h>

定义函数 void free(void \*ptr);

函数说明 参数ptr为指向先前由malloc()、calloc()或realloc()所返回的内存指针。调用free()后ptr所指的内存空间便会被收回。假若参数ptr所指的内存空间已被收回或是未知的内存地址，则调用free()可能会有无法预期的情况发生。若参数ptr为NULL，则free()不会有任何作用。

4.3 calloc（配置内存空间）

表头文件 #include <stdlib.h>

定义函数 void \*calloc(size\_t nmemb，size\_t size);

函数说明 calloc()用来配置nmemb个相邻的内存单位，每一单位的大小为size，并返回指向第一个元素的指针。这和使用下列的方式效果相同:malloc(nmemb\*size);不过，在利用calloc()配置内存时会将内存内容初始化为0。

返回值 若配置成功则返回一指针，失败则返回NULL。

1. 其他

getpagesize（取得内存分页大小）

表头文件 #include<unistd.h>

定义函数 size\_t getpagesize(void);

函数说明 返回一分页的大小，单位为字节（byte）。此为系统的分页大小，不一定会和硬件分页大小相同。

返回值 内存分页大小。附加说明在Intel x86 上其返回值应为4096bytes。

### 实验内容:

1. 分别使用命令和/proc文件系统列出系统当前内存的使用情况。
2. 启动几个耗时较长的后台进程（多个grep），分别使用free和vmstat连续实时观察内存的使用情况。
3. 用size工具观察三个不同的可执行文件的大小以及它们段的大小。
4. 启动一个耗时较长的后台进程，通过/proc文件系统查看该进程所有内存使用相关信息，并列出。
5. 编写一个程序，打印系统的页面大小。
6. 阅读并编译运行以下程序，总结内存映象文件的使用方法。

范例 /\* 利用mmap()来读取/etc/passwd 文件内容\*/

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<unistd.h>

#include<sys/mman.h>

main()

{

int fd;

void \*start;

struct stat sb;

fd=open(“/etc/passwd”,O\_RDONLY); /\*打开/etc/passwd\*/

fstat(fd,&sb); /\*取得文件大小\*/

start=mmap(NULL,sb.st\_size,PROT\_READ,MAP\_PRIVATE,fd,0);

if(start= = MAP\_FAILED) /\*判断是否映射成功\*/

return;

printf(“%s”,start);

munmap(start,sb.st\_size); /\*解除映射\*/

closed(fd);

}

1. 编写一个程序，利用内存映象文件，实现less工具的功能（多屏显示）。

### 实验报告：

完成上述要求中的第7题并提交报告。