**深 圳 大 学 实 验 报 告**

**课程名称： 操作系统**

**实验项目名称： 实验3 内存管理**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 张滇**

**报告人： 梁润宇 学号： 2021220003 班级： 2班**

**实验时间： 2023年5月4日-2023年5月31日**

**实验报告提交时间： 2024年5月30日**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **1.实验目的：**  1.加深对内存分配与使用操作的直观认识；  2.掌握Linux操作系统的内存分配与使用的编程接口；  3.了解Linux操作系统中进程的逻辑编程地址和物理地址间的映射；  **2.实验要求：**  操作部分（参考）：（90分）   1. 借助google工具查找资料，学习使用Linux进程的内存分配、释放函数； 2. 借助google工具查找资料，学习Linux proc文件系统中关于内存影射的部分内容（了解/proc/pid/目录下的maps、status、smap等几个文件内部信息的解读）； 3. 编写程序，连续申请分配六个128MB空间（记为1~6号），然后释放第2、3、5号的128MB空间。然后再分配1024MB，记录该进程的虚存空间变化（/proc/pid/maps），每次操作前后检查/proc/pid/status文件中关于内存的情况，简要说明虚拟内存变化情况。推测此时再分配64M内存将出现在什么位置，实测后是否和你的预测一致？解释说明用户进程空间分配属于课本中的离散还是连续分配算法？首次适应还是最佳适应算法？用户空间存在碎片问题吗？ 4. 设计一个程序测试出你的系统单个进程所能分配到的最大虚拟内存空间为多大。 5. 编写一个程序，分配256MB内存空间（或其他足够大的空间），检查分配前后/proc/pid/status文件中关于虚拟内存和物理内存的使用情况，然后每隔4KB间隔将相应地址进行读操作，再次检查/proc/pid/status文件中关于内存的情况，对比前后两次内存情况，说明所分配物理内存（物理内存块）的变化。然后重复上面操作，不过此时为写操作，再观察其变化。   提高部分：（10分）  1） 编写并运行（在第5步的程序未退出前）另一进程，分配等于或大于物理内存的空间，然后每隔4KB间隔将相应地址的字节数值增1，此时再查看前一个程序的物理内存变化，观察两个进程竞争物理内存的现象。  2） 分配足够大的内存空间，其容量超过系统现有的空闲物理内存的大小，1）按4KB的间隔逐个单元进行写操作，重复访问数遍（使得程序运行时间可测量）；2）与前面访问总量和次数不便，但是将访问分成16个连续页为一组，逐组完成访问，记录运行时间。观察系统的状态，比较两者运行时间，给出判断和解释。  **3.实验报告要求：**   1. 按学校统一格式 2. 需要给出具体命令和自行编写的程序的源代码 3. 程序的设计需要给出设计思路或流程框图 4. 实验操作的截图需要有必要的说明文字 |

|  |
| --- |
| **实验过程：**  1.借助google工具查找资料，学习使用Linux进程的内存分配、释放函数  图1-google搜索Linux进程的内存分配与释放  下面对一些常用的函数进行介绍。   1. malloc   malloc（memory allocation）用于从堆中分配内存。它接受一个参数，指定要分配的字节数，并返回指向分配内存块的指针。如果分配失败，它会返回NULL。   1. free   free用于释放之前由malloc、calloc或realloc分配的内存块。释放内存后，指针不再有效，应避免使用。   1. calloc   calloc（contiguous allocation）用于分配一个内存块并将其初始化为零。它接受两个参数：元素个数和每个元素的大小。   1. realloc   realloc（reallocation）用于调整已分配内存块的大小。它可以扩大或缩小内存块，并在必要时移动内存块。   1. brk和 sbrk   这两个系统调用用于调整进程的数据段末尾，改变程序数据段的大小。brk直接设置数据段末尾，而sbrk增加或减少数据段大小。   1. mmap和 munmap   mmap用于映射文件或设备到内存，或者分配匿名内存（即不与文件关联的内存），而munmap用于解除映射。   1. mprotect   mprotect用于改变已映射内存区域的保护属性，例如使内存区域只读、只写或可执行。  2.借助google工具查找资料，学习Linux proc文件系统中关于内存影射的部分内容（了解/proc/pid/目录下的maps、status、smap等几个文件内部信息的解读）  (1) /proc/[pid]/maps  maps文件显示了进程的内存映射信息。每一行表示一个内存区域，包含以下字段：  address perms offset dev inode pathname  00400000-0040b000 r-xp 00000000 08:01 1234567 /usr/bin/cat  address：内存区域的起始地址和结束地址  perms：权限，分别表示读（r）、写（w）、执行（x）和私有/共享（p/s）  offset：在文件中的偏移量  dev：设备号  inode：文件的inode号  pathname：文件路径（如果是文件映射）  (2) /proc/[pid]/status  status文件包含了进程的状态信息，以可读的格式呈现。主要字段包括：  Name: cat  State: R (running)  Tgid: 1234  Pid: 1234  PPid: 1  ...  VmSize: 456 kB  VmRSS: 123 kB  Name：进程名称  State：进程状态（如R运行、S睡眠、D不可中断、Z僵尸等）  Tgid：线程组ID  Pid：进程ID  PPid：父进程ID  VmSize：进程虚拟内存总大小  VmRSS：进程实际占用的物理内存大小  (3) /proc/[pid]/smaps  smaps文件包含了每个内存映射的详细信息，比maps文件更详细。每个映射区域包含多个字段，描述其内存使用情况：  00400000-0040b000 r-xp 00000000 08:01 1234567 /usr/bin/cat  Size: 44 kB  Rss: 20 kB  Pss: 10 kB  Shared\_Clean: 12 kB  Shared\_Dirty: 0 kB  Private\_Clean: 8 kB  Private\_Dirty: 0 kB  Referenced: 20 kB  ...  Size：映射区域的大小  Rss：驻留集大小（实际使用的物理内存）  Pss：比例驻留集大小（共享内存按比例计算）  Shared\_Clean：共享的干净内存  Shared\_Dirty：共享的脏内存  Private\_Clean：私有的干净内存  Private\_Dirty：私有的脏内存  Referenced：被引用的内存大小  3.编写程序，连续申请分配六个128MB空间（记为1~6号），然后释放第2、3、5号的128MB空间。然后再分配1024MB，记录该进程的虚存空间变化（/proc/pid/maps），每次操作前后检查/proc/pid/status文件中关于内存的情况，简要说明虚拟内存变化情况。推测此时再分配64M内存将出现在什么位置，实测后是否和你的预测一致？解释说明用户进程空间分配属于课本中的离散还是连续分配算法？首次适应还是最佳适应算法？用户空间存在碎片问题吗？  图2-程序流程  首先，定义一个函数来打印maps和status文件。然后再申请前，申请后，释放后，分配后，继续分配后调用该函数。  图3-print\_memory\_info函数  在main函数内，进行内存分配和释放。每次操作前后，都输出当前内存情况。  图4-main函数过程  在分配内存前，heap区从地址60830f69c000-60830f6bd000，虚拟内存大小为2656kb  图5-申请前  分配内存后，多出来77197e9fa000-7719aea00000的内存（图6），换算后我们得到大小大约是768MB也就是6个128MB的内存+6×4096字节，总共也是805330944字节。  此时，虚拟内存变成了789112KB  图6-申请6块空间后  释放2,3,5缓冲区后，虚拟内存变成了395884KB，也就是减少了393228KB，大概348MB，也就是是3个128MB缓冲区释放的大小。  heap区不再是连续的内存，而是3块大小为134221824KB的内存块，大概128MB。  图7-释放2,3,5缓冲区后  再分配1024MB的空间，可以看到我们的这段空间的地址是77197e9fa000-7719aea00000，接着我们再查看maps，通过与之前对比我们发现1024MB的内存被扩展到了6号内存之后，也就是增加了0x1000的内存。  图8-申请1024MB之后  此时共享内存为144464KB。  如果此时再分配64M内存，很可能会出现在之前释放的128MB块的位置之一，也有可能被操作系统选择一个新的地址空间。  图9-申请64MB后  可以看到，此时64MB内存的位置在原2号内存处。  1. 用户进程空间分配属于课本中的离散还是连续分配算法？  用户进程空间分配在现代操作系统中主要使用的是离散分配算法，这种方法是基于分页和分段技术。  2. 首次适应还是最佳适应算法？  在分页和分段的离散分配系统中，内存分配的策略主要是由内存管理单元（MMU）和操作系统内核管理的，通常不直接采用简单的首次适应（First-Fit）或最佳适应（Best-Fit）算法。  3. 用户空间存在碎片问题吗？  存在两种碎片问题：  内部碎片：当分配的内存块比实际需要的内存稍大时，导致内存块内部有未使用的部分。这在分页系统中尤为常见，因为每个页的大小是固定的，如果进程需要的内存量不是页大小的整数倍，就会有部分内存被浪费。  外部碎片：当内存中存在足够总量的空闲空间，但这些空闲空间不是连续的，无法满足某个较大内存请求。尽管现代分页系统通过离散分配方式缓解了外部碎片问题，但在一些特定分段系统或内存池中仍可能存在。  4.设计一个程序测试出你的系统单个进程所能分配到的最大虚拟内存空间为多大。  为了设计一个程序测试系统单个进程所能分配的最大虚拟内存空间，可以逐步增加内存分配的大小，直到分配失败为止。  使用 malloc 函数进行内存分配，并在每次分配时检查是否成功，并记录已分配的总内存量。  图10-寻找可分配内存上限  最后得到一个进程能分配的虚拟内存上限为131061GB。  图11-虚拟内存上限结果  对于64位系统，理论上可以支持2^64个字节，即 16 EB。但操作系统本身会对虚拟地址空间进行限制。例如Linux和Windows等操作系统会对用户态进程和内核态进程进行地址空间分隔，并且操作系统内核可能不支持使用所有的地址空间。  在Linux系统中，常见的配置是用户态进程最多可以使用128 TB的虚拟地址空间（48位地址空间）。  我使用的Ubuntu虚拟机可分配的虚拟内存最大为131061GB，接近128 TB（131072GB）。这说明我的虚拟机环境配置了128TB的用户态进程虚拟地址空间。  5.编写一个程序，分配256MB内存空间（或其他足够大的空间），检查分配前后/proc/pid/status文件中关于虚拟内存和物理内存的使用情况，然后每隔4KB间隔将相应地址进行读操作，再次检查/proc/pid/status文件中关于内存的情况，对比前后两次内存情况，说明所分配物理内存（物理内存块）的变化。然后重复上面操作，不过此时为写操作，再观察其变化。  print\_memory\_info函数和任务3中是一样的，这里不再重复。  编写touch\_memory函数执行4KB的读操作和写操作。  图12-touch\_memory函数  图13-主函数内容  主函数内就分配内存，查看内存信息，读操作、查看内存信息，写操作、查看内存信息。  1. 初始状态  图14-初始状态  VmSize : 2656 kB。这表示进程的虚拟内存总量，包括代码段、数据段、堆、栈和共享库等。  VmRSS: 1536 kB。这表示实际驻留在物理内存中的页面数量。  VmData : 224 kB。这表示数据段（包括堆）使用的内存量。  2. 分配256MB内存后  图15-分配256MB内存后  VmSize 增加到了 264804 kB，增加了约 256MB。这是因为调用 malloc 分配了256MB的虚拟内存。  VmRSS 没有变化，仍然是 1536 kB。这表明虽然虚拟内存增加了，但实际物理内存还没有被分配，因为 malloc 只是预留虚拟地址空间，并没有实际访问这些内存。  VmData 增加到了 262372 kB，这表明数据段的大小增加了256MB。  3. 读操作后  图16-读操作后  VmSize 仍然是 264804 kB，没有变化，因为虚拟内存总量保持不变。  VmRSS 仍然是 1536 kB，没有变化。这表明读操作没有导致物理内存的实际分配，因为读操作不会触发缺页中断，数据页仍然没有实际载入物理内存。  VmData 仍然是 262372 kB，没有变化。  4. 写操作后  图17-写操作后  VmSize 仍然是 264804 kB，没有变化。  VmRSS 增加到了 263680 kB，增加了约 256MB。这表明写操作触发了缺页中断，导致实际物理内存被分配并写入数据。  VmData 仍然是 262372 kB，没有变化。  5. 释放内存后  图18-释放内存后  VmSize 恢复到了 2656 kB，回到了初始状态。这表明释放了256MB的虚拟内存。  VmRSS 降到了 1632 kB，略高于初始值的 1536 kB。这可能是由于程序其他部分的内存开销。  VmData 恢复到了 224 kB，回到了初始状态。  结果分析：  1.分配256MB内存后：虚拟内存显著增加，但物理内存没有变化。  2.读操作后：虚拟和物理内存都没有显著变化，说明读操作没有触发物理内存分配。  3.写操作后：物理内存显著增加，表明写操作触发了缺页中断，导致实际分配物理内存。  ++++++++++++++++++++++++++++++++++++++++++++++++++++++  **其他（例如感想、建议等等）。**  感想：本次实验在对内存的分配和释放过程中，观测虚拟内存和物理内存的使用情况，对操作系统的内存管理有了更深的理解。 |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：张滇  2024年 4 月 30 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。