Proj 4：内存管理

> 现场验收时间：2025.6.15晚上18:00-21:00，实验楼103
>
> 报告提交截止时间：验收通过后，2025.6.17晚上22:00

本实验探讨内存管理技术。

系统启动后，空闲页框数会显示：

```
xv6...
cpu0: starting 0
number of free frames: 56932
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start
58
init: starting sh
```

1.堆空间的延迟页面分配 （70%）

xv6中，应用程序通过系统调用向内核申请堆空间，该系统调用为sbrk（例如malloc，即先使用sbrk申请空间后对该空间进行管理）。调用sbrk后，即使该空间未必会被使用，内核也会立即为应用程序分配物理内存。请你修改xv6的行为，使其在应用程序实际上访问某个地址时，才相应地分配物理内存。

没有做任何修改时，运行memtest，有2个测试可以通过，Test 4 和 Test 5，在你做实验的过程中，为了通过Test 1-3，可能会导致Test 4和Test 5无法通过；建议边读memtest.c的内容，边修改。

```
$ memtest
number of free frames: 56789
Test 1 (sbrk without allocating memory):  fail2!
Test 2 (write to heap mem):  fail!
Test 3 (deallocating memory):  fail!
Test 4 (allocating too much mem):  success!
Test 5 (access invalid mem, two page faults):
pid 67 memtest: trap 14 err 6 on cpu 0 eip 0x235 addr 0x7000--kill proc
pid 68 memtest: trap 14 err 5 on cpu 0 eip 0x26d addr 0x1fff--kill proc
```

实验内容

- 应用程序申请空间时，假装已分配，但实际不分配物理内存（类似于请求分页）。
  - 你需要修改sbrk函数，不分配物理内存，但正常返回，让应用程序认为已分配了内存。
  - sbrk中，若传入的参数过大（参考growproc函数），则返回-1，表示请求失败。
  - sbrk中，若传入的参数为负值，则立即释放内存，参考growproc（允许直接调用这个函数）。
- 应用程序访问该空间时，若尚未分配物理内存，则触发缺页中断，此时分配物理内存。
  - 你需要扩展trap函数，捕获并处理缺页中断，请你添加：

```
case T_PGFLT:
    myproc()->tf = tf;
    pgflt_hdl();  // page fault handler
    break;
```

  - 你需要实现pgflt_hdl函数，用于处理缺页中断。
  - pgflt_hdl中，每次只能分配一页（与growproc不同，growproc一次分完）。
  - pgflt_hdl中，若触发缺页中断的地址非法（如超过堆的最高地址），则报错并杀死进程。
  - pgflt_hdl中，你需要适时刷新快表。你可以通过汇编代码重置cr3寄存器，或使用switchuvm(myproc())。
- 当部分地址未分配物理内存时，fork可能会出问题，此时检查copyuvm的实现，里面有一些句子（不多）需要修改（Test 5）。
- 当应用程序访问未分配的堆空间地址，即非法访问，xv6应当正常报错，而不是分配物理内存（Test 5）。

实验提示

- 本质上是将growproc的物理内存的分配延迟到中断处理中，请参考growproc。
- xv6的内存保护的粒度为页面，某些错误可能无法触发缺页中断而漏报（如访问myproc()->sz到上页对齐地址之间的地址），无须额外处理。
- 整个实验通过调用growproc，allocuvm可以完成，当然也可以重现两个函数的具体做法。

验收要求

- 运行memtest，预期输出：

```
$ memtest
number of free frames: 56789
Test 1 (sbrk without allocating memory):  success!
Test 2 (write to heap mem):  success!
Test 3 (deallocating memory):  success!
Test 4 (allocating too much mem):  success!
Test 5 (access invalid mem, two page faults):
pid 5 memtest: trap 6 err 0 on cpu 569 eip 0x7000 addr 0x8df76f78--kill proc
pid 6 memtest: trap 5 err 0 on cpu 625 eip 0x1fff addr 0x8df75f78--kill proc
```

  - 第一行打印的空闲页框数可能不是56789。
  - 第一行打印的空闲页框数在多次memtest中相同（否则存在物理内存未释放）。
  - Test 5的报错信息可以自定义，信息量相近即可。
- 运行forktest，预期输出：

```
$ forktest
fork test
fork test OK
```

报告要求

- 介绍你的实现，说明困难或者问题（如果有）。

2. 写时复制机制的设计（10%）

此部分不需要实现。

调查现代Linux系统中写时复制机制(copy-on-write，COW)的实现机制。如果在xv6中实现写时复制的fork，需要做哪些修改？写到实验报告中。

3.实验报告和代码（20%）

在助教现场验收结束后，需要准备两份文件：

- 源代码：zip压缩包
  - 进入`proj4-revise`目录，执行`make clean`，然后在proj4-revise的父目录下执行`zip -r proj4-revise.zip proj4-revise`，此命令会创建一个源代码压缩文件`proj4-revise.zip`。提交此文件。
  - 请确保提交的代码压缩包内无其他文件（比如.o文件，或者vscode目录）。
- 实验报告：pdf，
  - 在实验报告中回答上面提到的问题；
  - 给出参考资料（网址）和工具（哪个大模型）；
  - 报告中需要添加姓名学号。不限定模板，尽量整洁、美观。