哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验设计报告

开课学期：	2022 年秋季
课程名称：	操作系统
实验名称：	Lazy
实验性质：	额外实验
实验时间：	10.30　地点：T2
学生班级：	11
学生学号：	200111132
学生姓名：	吴桐
评阅教师：	
报告成绩：	

实验与创新实践教育中心印制

2022 年 9 月

一、 实验详细设计

注意不要照搬实验指导书上的内容，请根据你自己的设计方案来填写

主要思路：在 sbrk 的时候并不为用户态分配内存。知道需要的时候触发 pagefault 的时候才会分配实际的内存页

Sysproc.c

修改了 sys_sbrk，本身在 n>0 的时候要分配新的页，但是这里就只修改 sz。

Trap.c

在这里 usertrap 的函数里，进行判断的时候多加一个 else if
判断原因是否是读写缺页异常（13 or 15），如果是进行该页的一个分配操作，具体操作我就写在下面的 vm 里面

Vm.c

新增了一个函数 lazyalloc(uint64 addr)

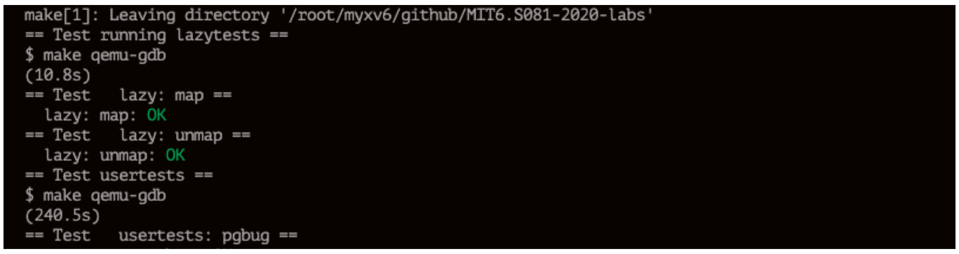
用于为缺页的部分分配实际内存页。首先我们需要检查需要分配的地址是否满足合法范围（`addr >= p->sz||addr<p->trapframe->sp`）
如果符合，就可以为它 kalloc 分配一个页，并且映射到页表里面。

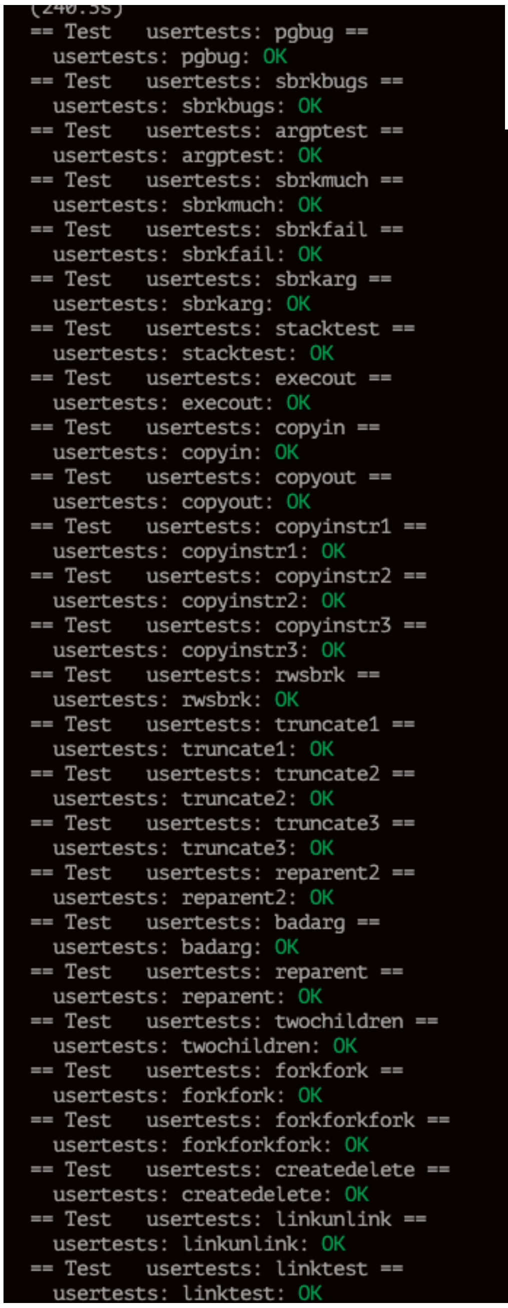
这里由于使用了 la 机制，所以之前部分函数的机制也需要修改：

Walkaddr：在搜索 pte 的时候，由于没有映射实际页，所以新的 pte 也找不到，并且不会触发缺页异常。这里我们找不到 pte 的时候需要先尝试 lazyalloc(va)一下，如果是合法的话，则再次 walk 就可以找到对应 pte。

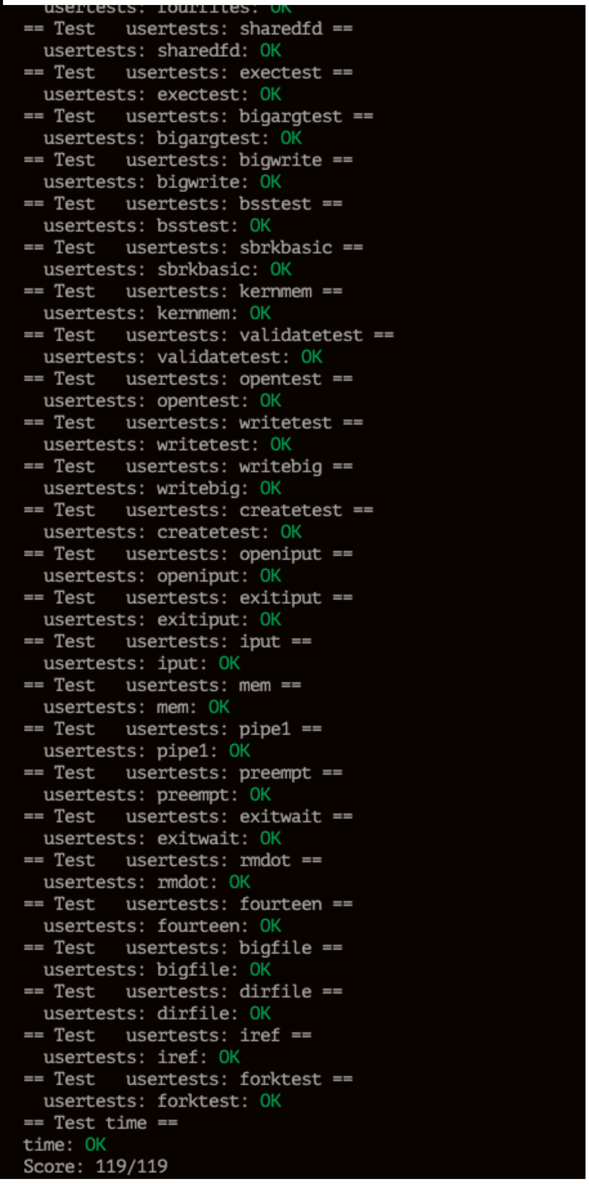
Uvmunmap: 同理，这里如果解映射的话没找到会 panic，但是有可能是我们根本就没映射到（换句话来说这个页就没啥用），如果没有解到，就可以直接 continue。

二、 实验结果截图


```
make[1]: Leaving directory '/root/myxv6/github/MIT6.S081-2020-labs'
== Test running lazytests ==
$ make qemu-gdb
(10.8s)
== Test   lazy: map ==
  lazy: map: OK
== Test   lazy: unmap ==
  lazy: unmap: OK
== Test usertests ==
$ make qemu-gdb
(240.5s)
== Test   usertests: pgbug ==
```



```
(240.3s)
== Test   usertests: pgbug ==
  usertests: pgbug: OK
== Test   usertests: sbrkbugs ==
  usertests: sbrkbugs: OK
== Test   usertests: argptest ==
  usertests: argptest: OK
== Test   usertests: sbrkmuch ==
  usertests: sbrkmuch: OK
== Test   usertests: sbrkfail ==
  usertests: sbrkfail: OK
== Test   usertests: sbrkarg ==
  usertests: sbrkarg: OK
== Test   usertests: stacktest ==
  usertests: stacktest: OK
== Test   usertests: execout ==
  usertests: execout: OK
== Test   usertests: copyin ==
  usertests: copyin: OK
== Test   usertests: copyout ==
  usertests: copyout: OK
== Test   usertests: copyinstr1 ==
  usertests: copyinstr1: OK
== Test   usertests: copyinstr2 ==
  usertests: copyinstr2: OK
== Test   usertests: copyinstr3 ==
  usertests: copyinstr3: OK
== Test   usertests: rwsbrk ==
  usertests: rwsbrk: OK
== Test   usertests: truncate1 ==
  usertests: truncate1: OK
== Test   usertests: truncate2 ==
  usertests: truncate2: OK
== Test   usertests: truncate3 ==
  usertests: truncate3: OK
== Test   usertests: reparent2 ==
  usertests: reparent2: OK
== Test   usertests: badarg ==
  usertests: badarg: OK
== Test   usertests: reparent ==
  usertests: reparent: OK
== Test   usertests: twochildren ==
  usertests: twochildren: OK
== Test   usertests: forkfork ==
  usertests: forkfork: OK
== Test   usertests: forkforkfork ==
  usertests: forkforkfork: OK
== Test   usertests: createdelete ==
  usertests: createdelete: OK
== Test   usertests: linkunlink ==
  usertests: linkunlink: OK
== Test   usertests: linktest ==
  usertests: linktest: OK
```



```
  usertests: fourfiles: OK
== Test   usertests: sharedfd ==
  usertests: sharedfd: OK
== Test   usertests: exectest ==
  usertests: exectest: OK
== Test   usertests: bigargtest ==
  usertests: bigargtest: OK
== Test   usertests: bigwrite ==
  usertests: bigwrite: OK
== Test   usertests: bsstest ==
  usertests: bsstest: OK
== Test   usertests: sbrkbasic ==
  usertests: sbrkbasic: OK
== Test   usertests: kernmem ==
  usertests: kernmem: OK
== Test   usertests: validatetest ==
  usertests: validatetest: OK
== Test   usertests: opentest ==
  usertests: opentest: OK
== Test   usertests: writetest ==
  usertests: writetest: OK
== Test   usertests: writebig ==
  usertests: writebig: OK
== Test   usertests: createtest ==
  usertests: createtest: OK
== Test   usertests: openiput ==
  usertests: openiput: OK
== Test   usertests: exitiput ==
  usertests: exitiput: OK
== Test   usertests: iput ==
  usertests: iput: OK
== Test   usertests: mem ==
  usertests: mem: OK
== Test   usertests: pipe1 ==
  usertests: pipe1: OK
== Test   usertests: preempt ==
  usertests: preempt: OK
== Test   usertests: exitwait ==
  usertests: exitwait: OK
== Test   usertests: rmdot ==
  usertests: rmdot: OK
== Test   usertests: fourteen ==
  usertests: fourteen: OK
== Test   usertests: bigfile ==
  usertests: bigfile: OK
== Test   usertests: dirfile ==
  usertests: dirfile: OK
== Test   usertests: iref ==
  usertests: iref: OK
== Test   usertests: forktest ==
  usertests: forktest: OK
== Test time ==
time: OK
Score: 119/119
```

（我也不大清楚为啥这次跑出来全部 usertests 会打印出来🤣）