1、代码修改

- 合并代码并修改框架原有bug，将大部分printf改成debug防止过多输出
- 修改了sys_write和sys_gettimeofday，具体来说，sys_write改成fd不为1时return -1；sys_gettimeofday为适应虚实映射，首先设置一个局部变量TimeVal并给其赋值，再用copyout给参数赋值
- 完成了sys_mmap和sys_munmap，实现的方式为：
    - 1、检查参数是否满足条件；
    - 2、将len上取整并计算出要操作的页数；
    - 3、使用for循环遍历每一页：对于mmap，则用useraddr==0确认页未被映射，再用kalloc和mappages完成分配和映射；对于munmap，则用useraddr()!=0确认页已被映射，再用uvmunmap删除该页。

2、思考题

1、请列举 SV39 页表页表项的组成，结合课堂内容，描述其中的标志位有何作用 / 潜在作用?

63 54	53 28	27 19	18 10	9 8	7	6	5	4	3	2	1	0
Reserved	PPN[2]	PPN[1]	PPN[0]	RSW	D	A	G	U	X	W	R	V
10	26	9	9	2	1	1	1	1	1	1	1	1

上图为 SV39 分页模式下的页表项，其中 [53:10] 这 44 位是物理页号，最低的 88 位 [7:0][7:0] 则是标志位，它们的含义如下：

- 仅当 V(Valid) 位为 1 时，页表项才是合法的；
- R/W/X 分别控制索引到这个页表项的对应虚拟页面是否允许读/写/取指；
- G 表示是否为全局映射，在本操作系统中用不到；
- U 控制索引到这个页表项的对应虚拟页面是否在 CPU 处于 U 特权级的情况下是否被允许访问；
- A(Accessed) 记录自从页表项上的这一位被清零之后，页表项的对应虚拟页面是否被访问过；
- D(Dirty) 则记录自从页表项上的这一位被清零之后，页表项的对应虚拟页表是否被修改过。

2、缺页

1）请问哪些异常可能是缺页导致的?

代码运行时缺页，加载代码时试图读/写的时候缺页。

2）发生缺页时，描述相关的重要寄存器的值（lab2中描述过的可以简单点）。

3）这样做有哪些好处?

比较方便，当程序过大时，Lazy策略可以使得不在初始化的时候加载很久，而是在需要的时候才进行加载。

4）请问处理 10G 连续的内存页面，需要操作的页表实际大致占用多少内存(给出数量级即可)?

10G / 512 = 20M

5）请简单思考如何才能在现有框架基础上实现 Lazy 策略，缺页时又如何处理？描述合理即可，不需要考虑实现。

用户申请内存时，进行标记但不执行实际的内存分配，在用户进行读/写时再分配内存。缺页时分配实际内存

6）此时页面失效如何表现在页表项(PTE)上？

通过PTE_V，即页面不valid

3、双页表与单页表

1）如何更换页表？

通过satp切换。

2）单页表情况下，如何控制用户态无法访问内核页面？（tips:看看上一题最后一问）

将PTE_U置为0

3）单页表有何优势？（回答合理即可）

节约空间开销，实现方便。

4）双页表实现下，何时需要更换页表？假设你写一个单页表操作系统，你会选择何时更换页表（回答合理即可）？

当用户态/内核态切换的时候需要更换页表，以及用户态的不同程序切换的时候更换页表。会在用户态与内核态切换的时候更换页表。