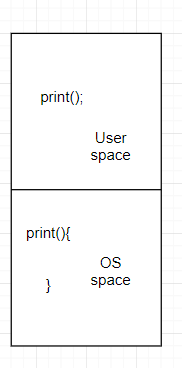
user process必须控制在user space内，OS必须有整个main memory的权限，不然他没法get到需要打印的file

mode bit:0 是user process,1是operating system

 因为没OS的权限，所以只能

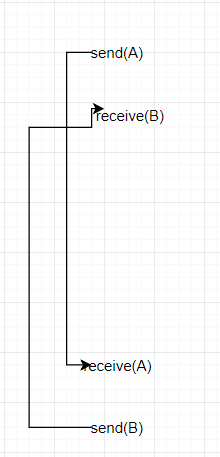
1.change power

当把mode bit改成1，就能激活OS中的完整method，读到最后一个花括号，改为0

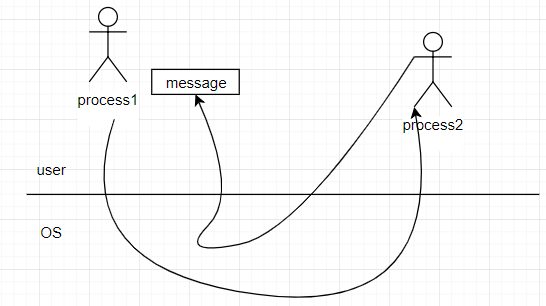
缺陷:不是很security,当OS的详细版本print()中有bug，就会crash,那么就到不了最后一个花括号，那么就一直有了1的power

2.转换成message，传递给OS中间人，生成新message，传递回去

缺陷：慢

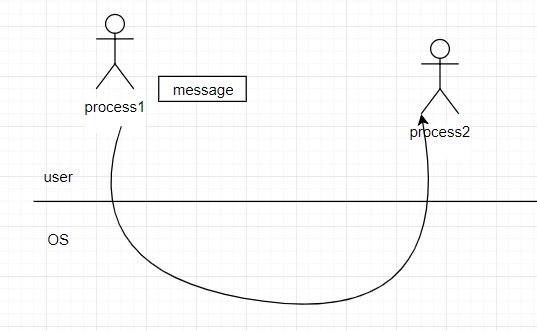


shared memory



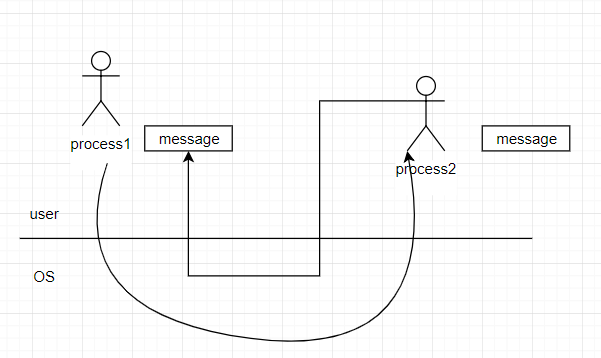
92％：发出idication（一个指令，告诉你我有信息，你能接受吗），然后user process就可以直接看 message

3％，receiver delete, 我阅读了，不想要，



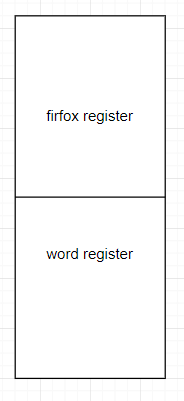
箭头删掉

2％，一方想要改变message，那么另一方就生成备份，删掉箭头，改变自己的（复制一次）



3％，spying

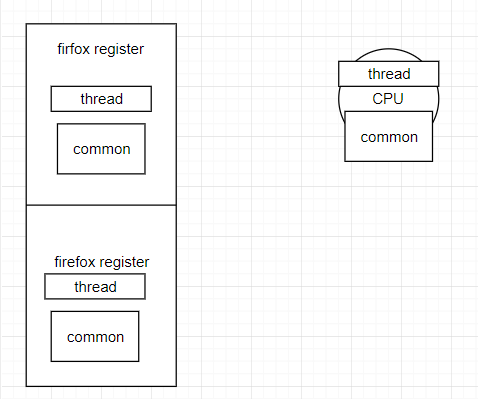
我发出了indication,还没等到user process回复我想要不想要，而我想改这个message，就只能把这个message复制给OS，然后user process再从process复制（复制两次）

运行firefox：CPU读取firefox的200 register，存储当前CPU内WORD的200个register

运行word:CPU读取word的200个register，存储对当前FIREFOX的200个register

一次运行有400个IO操作，太慢

两个firefox有同样的code，不同的部分用thread表示，

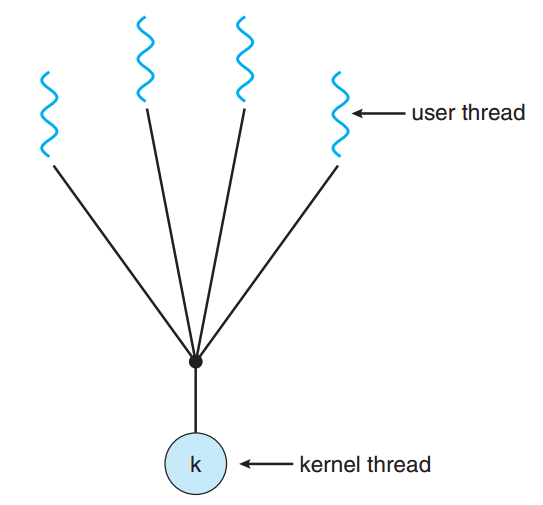


CPU keep 20到200的common部分，只switch1到20的thread部分，只用40个IO操作，

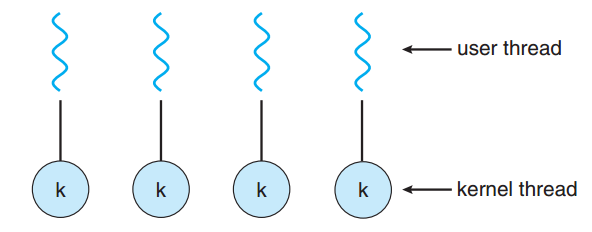
Process:完全copy

thread:重复的不copy

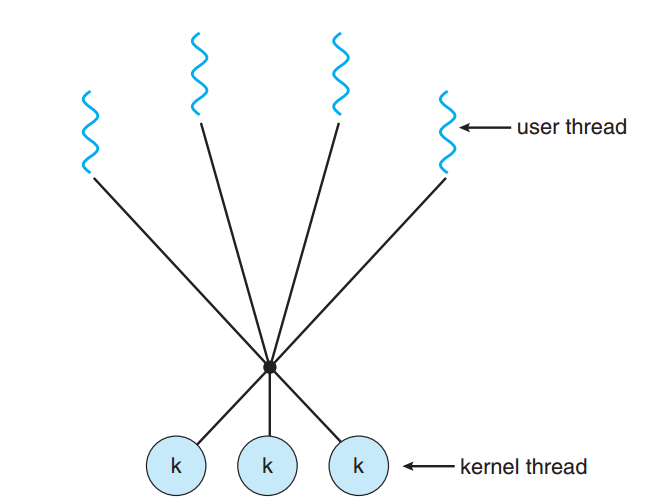
many to one model



one to one model



many to many model



不同的情况:

limited the kernel，不limit user

many to many>many to one>one to one

如果过kernel效率够高：many to one

kernel效率低，存储空间够大，one to one