**User-level threads**

1. 实验目的

通过实现代码来执行线程之间的上下文切换，从而完成一个简单的用户级线程包

二、实验内容

[Homework: User-level threads (mit.edu)](https://pdos.csail.mit.edu/6.828/2018/homework/xv6-uthread.html)

三、实验步骤与实验结论

（1）下载代码并编译运行，修改添加Makefile，UPROGS中添加\_uthread

\_uthread: uthread.o uthread\_switch.o

$(LD) $(LDFLAGS) -N -e main -Ttext 0 -o \_uthread uthread.o uthread\_switch.o $(ULIB)

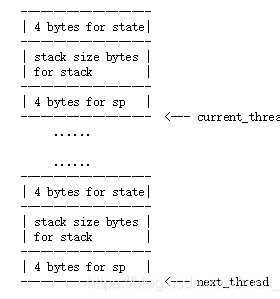
$(OBJDUMP) -S \_uthread > uthread.asm

（2）uthread程序会创建两个线程并在它们之间来回切换,每个线程都会打印"my thread…",然后放弃处理器让另一个线程运行。uthread.c有两个全局变量current\_thread和next\_thread.它们都是指向thread结构体的指针.thread结构体中有一个线程栈和一个保存的栈顶指针。

uthread\_switch的工作就是将current thread的状态保存到其对应的结构体中,然后恢复next\_thread的状态,并使current\_thread指向next\_thread所指向的结构体，以便当uthread\_switch返回时next\_thread正在运行并且是current\_thread。

thread\_create函数,该函数设置了新线程的初始栈并且给出了thread\_switch所做工作的提示,thread\_switch使用汇编指令popal和pushal还原并保存所有八个x86寄存器

（3）理解thread结构体以及线程指针位置



（4）ext\_thread和current\_thread 变量都保存了一个thread结构体的地址,将esp写入current\_thread->sp

movl current\_thread, %eax

movl %esp, (%eax)

（5）thread\_switch.S：curretn\_thread调用thread\_switch会自动将eip压入自己的堆栈,然后保存八个寄存器的值,之后修改current\_thread为next\_thread并切换堆栈,然后恢复现场,最后ret指令将(%eps-4)写入eip中以执行下一条指令



Q. 位于next\_thread堆栈顶部的0xd8是什么地址？

A. 是next\_thread在上一次让出处理机时保存的返回地址,也就是调用thread\_switch之后应当执行的下一条指令。

四、反思总结

.S 文件中可以直接使用 .c 文件中的变量