+ -------------------------- +

| CS 140 |

| 项目2：用户程序|

| 设计文件|

+ -------------------------- +

---- GROUP ----

>>填写小组成员的姓名和电子邮件地址。

名字姓氏<email@domain.example>

名字姓氏<email@domain.example>

名字姓氏<email@domain.example>

----前提----

>>如果您对提交的内容有任何初步意见，请注意该

>>助教，或额外的信用，请在这里给他们。

>>请引用您在咨询期间参考的所有离线或在线资源

>>准备您的提交内容，而不是Pintos文档，课程

>>文本，讲义和课程人员。

参数传递

===============

---- ----数据结构----

怎么写请见：[http://web.stanford.edu/~ouster/cgi-bin/cs140-spring20/pintos/pintos\_1.html#SEC8](http://web.stanford.edu/~ouster/cgi-bin/cs140-spring20/pintos/pintos_1.html" \l "SEC8)

http://web.stanford.edu/~ouster/cgi-bin/cs140-spring20/pintos/pintos\_9.html#SEC142

process.c

>> A1：在此处复制每个新的或更改的`struct'或

>>`struct'的声明成员，全局或静态变量，“ typedef”或

>>枚举。用25个字以内的单词来确定每个单词的目的。

----算法----

>> A2：简要描述如何实现参数解析。

>>您如何安排argv []的元素以正确的顺序排列？

>>如何避免堆栈页面溢出？

如何实现参数传递：在process.c的start\_process()函数中使用结构体intr\_frame，用堆栈的指针实现。

argv[]的顺序：栈顶指针esp是不断减小的，也就是从后往前获取。用这种方法就能依次获取argv,argc和返回地址。

避免堆栈页面溢出：esp指针不合法直接退出进程。

----理由----

>> A3：为什么Pintos实现strtok\_r（）但不实现strtok（）？

strtok()函数将每次调用后的字符串位置保存在一个函数内部的静态局部变量中，如果有多个线程调用这个函数，有可能出现冲突。

而strtok\_r()使用自定义指针指向切好的字符串，是线程安全的

>> A4：在Pintos中，内核将命令分为可执行文件名

>>和参数。在类似Unix的系统中，shell进行

>>分离。至少确定Unix方法的两个优点。

优点：1.分离命令可以用管道 2.可以检查该文件是否存在（用文件名）

系统调用

===========

syscall.c

---- ----数据结构----

>> B1：在此复制每个新的或更改的`struct'或

>>'struct'成员的声明，全局或静态变量`typedef'或

>>枚举。用25个字以内的单词来确定每个单词的目的。

>> B2：描述文件描述符如何与打开的文件关联。

>>用syscall.c打开的文件的文件描述符都是一一对应的。

>>文件描述符在整个操作系统中还是在单个进程中是唯一的？

文件描述符在整个操作系统中是唯一的。在syscall.c中建立了一个文件链表file\_list，而不是在单独的进程中。

----算法----

>> B3：描述用于从

>>内核读取和写入用户数据的代码。

READ：先检查指针fd,buffer,size是否有效（无效直接退出）。获取锁并调用read()函数。在read()函数中，如果是 (fd == STDIN)就是键盘输入，要不就是用fd编号获取文件，判断文件指针是否为空，非空就调用file\_read()并返回状态。最后释放锁。

WRITE：先检查指针fd,buffer,size是否有效（无效直接退出），再检查buffer指针是否有效。获取锁并调用write()函数。在write()函数中，如果是 (fd == STDOUT)就是写入控制台，要不就是用fd编号获取文件，判断文件指针是否为空，非空调用file\_write()从缓冲区写入文件并返回状态。最后释放锁。

>> B4：假设系统调用导致整页（4,096字节）的数据从用户空间复制到内核。 可能导致的对页表的检查次数最少（例如最大）（例如，对pagedir\_get\_page（）的调用）？ 对于仅复制2个字节的数据的系统调用该怎么办？ 这些数字是否还有改进的余地，还有多少？

整页数据：次数最少为1。不连续数据最大为4096.

2字节：最少为1，最大为2

>> B5：简要描述“ wait”系统调用的实现

>>以及它如何与流程终止交互。

直接调用process\_wait()。

需要判断是不是调用者的孩子，如果不是就终止。使用is\_child()函数判断。

>> B6：

由于错误的指针值，对用户指定地址处的用户程序存储器的任何访问都可能失败。 这样的访问必须导致进程终止。 系统调用中充斥着此类访问，例如 “写入”系统调用需要从用户堆栈中读取系统调用号，然后是该调用的三个参数中的每个参数，然后是任意数量的用户内存，任何这些都可能在任何时候失败。 这带来了设计和错误处理问题：如何最好地避免在错误处理的混乱中模糊代码的主要功能？ 此外，当检测到错误时，如何确保释放所有临时分配的资源（锁，缓冲区等）？ 在几段中，描述您用于管理这些问题的策略。 举个例子。

如何避免：

例如：read操作。

函数会先检查是否为无效指针（调用检查函数），然后检查缓冲区的开始和结束指针是否无效。检测到错误时，调用exit()函数直接关闭fd对应文件并清空文件链表。

---- SYNCHRONIZATION同步 ----

>> B7：如果加载新的可执行文件

>>失败，“ exec”系统调用将返回-1 ，因此在新的可执行文件完成

>>加载之前，它无法返回。您的代码如何确保这一点？负载

>>成功/失败状态如何传递回调用“ exec”的线程？

>> B8：将父进程P与子进程C一起考虑。

当P

>>时，如何确保正确的同步并避免争用条件 在C退出之前调用wait（C）？在C退出之后？您如何确保

>>在每种情况下都释放了所有资源？

在C退出之前P >>何时终止而没有等待又如何呢？在C退出之后？是否

>>有什么特殊情况？

---- RATIONALE 理由----

>> B9：为什么选择以

这种方式从>>内核实现对用户内存的访问？

>> B10：

对于文件描述符，您可以看到设计的哪些优缺点？

>> B11：默认的tid\_t到pid\_t映射是身份映射。

>>如果您进行了更改，您的方法有什么优势？

+--------------------------+

| CS 140 |

| PROJECT 2: USER PROGRAMS |

| DESIGN DOCUMENT |

+--------------------------+

---- GROUP ----

>> Fill in the names and email addresses of your group members.

FirstName LastName <email@domain.example>

FirstName LastName <email@domain.example>

FirstName LastName <email@domain.example>

---- PRELIMINARIES ----

>> If you have any preliminary comments on your submission, notes for the

>> TAs, or extra credit, please give them here.

>> Please cite any offline or online sources you consulted while

>> preparing your submission, other than the Pintos documentation, course

>> text, lecture notes, and course staff.

ARGUMENT PASSING

================

---- DATA STRUCTURES ----

>> A1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

---- ALGORITHMS ----

>> A2: Briefly describe how you implemented argument parsing. How do

>> you arrange for the elements of argv[] to be in the right order?

>> How do you avoid overflowing the stack page?

---- RATIONALE ----

>> A3: Why does Pintos implement strtok\_r() but not strtok()?

>> A4: In Pintos, the kernel separates commands into a executable name

>> and arguments. In Unix-like systems, the shell does this

>> separation. Identify at least two advantages of the Unix approach.

SYSTEM CALLS

============

---- DATA STRUCTURES ----

>> B1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

>> B2: Describe how file descriptors are associated with open files.

>> Are file descriptors unique within the entire OS or just within a

>> single process?

---- ALGORITHMS ----

>> B3: Describe your code for reading and writing user data from the

>> kernel.

>> B4: Suppose a system call causes a full page (4,096 bytes) of data

>> to be copied from user space into the kernel. What is the least

>> and the greatest possible number of inspections of the page table

>> (e.g. calls to pagedir\_get\_page()) that might result? What about

>> for a system call that only copies 2 bytes of data? Is there room

>> for improvement in these numbers, and how much?

>> B5: Briefly describe your implementation of the "wait" system call

>> and how it interacts with process termination.

>> B6: Any access to user program memory at a user-specified address

>> can fail due to a bad pointer value. Such accesses must cause the

>> process to be terminated. System calls are fraught with such

>> accesses, e.g. a "write" system call requires reading the system

>> call number from the user stack, then each of the call's three

>> arguments, then an arbitrary amount of user memory, and any of

>> these can fail at any point. This poses a design and

>> error-handling problem: how do you best avoid obscuring the primary

>> function of code in a morass of error-handling? Furthermore, when

>> an error is detected, how do you ensure that all temporarily

>> allocated resources (locks, buffers, etc.) are freed? In a few

>> paragraphs, describe the strategy or strategies you adopted for

>> managing these issues. Give an example.

---- SYNCHRONIZATION ----

>> B7: The "exec" system call returns -1 if loading the new executable

>> fails, so it cannot return before the new executable has completed

>> loading. How does your code ensure this? How is the load

>> success/failure status passed back to the thread that calls "exec"?

>> B8: Consider parent process P with child process C. How do you

>> ensure proper synchronization and avoid race conditions when P

>> calls wait(C) before C exits? After C exits? How do you ensure

>> that all resources are freed in each case? How about when P

>> terminates without waiting, before C exits? After C exits? Are

>> there any special cases?

---- RATIONALE ----

>> B9: Why did you choose to implement access to user memory from the

>> kernel in the way that you did?

>> B10: What advantages or disadvantages can you see to your design

>> for file descriptors?

>> B11: The default tid\_t to pid\_t mapping is the identity mapping.

>> If you changed it, what advantages are there to your approach?

SURVEY QUESTIONS

================

Answering these questions is optional, but it will help us improve the

course in future quarters. Feel free to tell us anything you

want--these questions are just to spur your thoughts. You may also

choose to respond anonymously in the course evaluations at the end of

the quarter.

>> In your opinion, was this assignment, or any one of the three problems

>> in it, too easy or too hard? Did it take too long or too little time?

>> Did you find that working on a particular part of the assignment gave

>> you greater insight into some aspect of OS design?

>> Is there some particular fact or hint we should give students in

>> future quarters to help them solve the problems? Conversely, did you

>> find any of our guidance to be misleading?

>> Do you have any suggestions for the TAs to more effectively assist

>> students, either for future quarters or the remaining projects?

>> Any other comments?