# 附录 F　实验室：系统项目

本章介绍了系统项目的一些想法。我们通常在为期 15 周的学期中完成 6～7 项目，这意味着每两周左右一个项目。前几个通常由学生独立完成，后几个通常是两人小组。

每个学期，项目都遵循同样的大纲。然而，我们会改变细节，让它保持有趣，这让跨学期的代码“共享”更有难度（并非任何人都会这样做！）。我们还使用 Moss 工具来检查这种“共享”。

至于评分，我们尝试了许多不同的方法，每种方法都有自己的优点和缺点。演示很有趣但很耗时。自动化测试脚本耗时较少，但需要非常谨慎，才能让它们仔细测试有趣的角落情况。查看本书的配套网页，了解有关这些项目的更多详情。如果你想要自动化测试脚本，我们很乐意分享。

## F.1　介绍项目

第一个项目是系统编程的介绍。典型的作业是编写 sort 实用程序的一些变体，加上不同的约束。例如，排序文本数据，排序二进制数据和其他类似项目都是有意义的。要完成项目，必须熟悉一些系统调用（及其返回错误代码），使用一些简单的数据结构，没有太多其他内容。

## F.2　UNIX Shell

在这个项目中，学生构建了 UNIX shell 的变体。学生将学习进程管理，以及管道和重定向等神秘事物的实际工作方式。变体包括不寻常的功能，例如一个重定向符号，它通过 gzip 压缩输出。另一种变体是批处理模式，它允许用户批量处理一些请求，然后执行它们，可能使用不同的调度规则。

## F.3　内存分配库

该项目通过构建一个替代的内存分配库（类似 malloc()和 free()，但具有不同的名称），来探索如何管理一块内存。该项目教会学生如何使用 mmap()获取一大块匿名内存，然后仔细使用指针，以构建一个简单（或可能较复杂）的空闲列表来管理空间。变体包括：最优/最差匹配、伙伴算法和各种其他分配器。

## F.4 并发简介

该项目引入了 POSIX 线程的并发编程。构建一些简单的线程安全库：列表、哈希表和一些更复杂的数据结构，是向现实代码添加锁的好练习。测量粗粒度与细粒度锁方案的性能。变体就是关注不同的（也许更复杂的）数据结构。

## F.5 并发 Web 服务器

该项目探索在实际应用中使用并发性。学生使用一个简单的 Web 服务器（或构建一个），并向其添加一个线程池，以便同时处理请求。线程池应该是固定大小的，并使用生产者/消费者有界缓冲区，将请求从主线程传递到固定的工作线程池。了解如何使用线程、锁和条件变量来构建真实服务器。变体包括线程的调度策略。

## F.6 文件系统检查器

该项目探讨了磁盘上的数据结构及其一致性。学生构建一个简单的文件系统检查器。debugfs 工具可以在 Linux 上用于制作真正的文件系统映像，检查它们，确保一切都正常。为了增加难度，还要修复发现的所有问题。变体关注不同类型的问题：指针、链接计数、间接块的使用等。

## F.7 文件系统碎片整理程序

该项目探讨了磁盘上的数据结构及其性能影响。该项目应该为学生提供一些特定的文件系统映像，它有已知的碎片问题。然后，学生应该检查该映像，并寻找未按顺序排列的文件。写出一个“消除碎片”的新映像来修复这个问题，可能要报告一些统计信息。

## F.8 并发文件服务器

该项目结合了并发和文件系统，甚至还有一些网络和分布式系统。学生构建一个简单的并发文件服务器。该协议应该看起来像 NFS，包括查找、读取、写入和状态信息。将文件存储在单个磁盘映像（设计为一个文件）中。变体是多种多样的，包括不同建议的磁盘格式和网络协议。

# 附录 G　实验室：xv6 项目

本章介绍了与 xv6 内核相关的项目的一些想法。该内核可从麻省理工学院获得，玩起来很有趣。完成这些项目还可以使课堂上的内容与项目更直接相关。这些项目（可能除了前面两个）通常是结对完成的，这让盯着内核代码的艰巨任务变得更加容易。

## G.1　简介项目

简介项目为 xv6 添加一个简单的系统调用。可能存在许多不同的任务，包括用一个系统调用来计算已发生的系统调用次数（每次系统调用计数一次），或其他信息收集的调用。学生将学习如何实际进行系统调用。

## G.2　进程和调度

学生构建比默认的轮询更复杂的调度程序。可能存在许多不同的策略，包括彩票调度程序或多级反馈队列。学生将学习调度程序的实际工作方式，以及上下文切换的方式。一个附加的小任务还要求学生弄清楚，如何在退出时让进程返回正确的错误代码，并能够通过 wait()系统调用访问该错误代码。

## G.3　虚拟内存简介

基本思想是添加一个新的系统调用，给定一个虚拟地址，返回已翻译的物理地址（或报告该地址无效）。这让学生可以看到虚拟内存系统如何设置页表而无需做太多艰苦的工作。另一个可能的项目是探索如何改动 xv6，让空指针引用会产生错误。

## G.4　写时复制映射

该项目为 xv6 增加轻量级 fork()的能力，名为 vfork()。这个新调用不是简单地复制映射，而是将写时复制映射设置为共享面。在引用这样的页面时，内核必须相应地创建真实副本并更新页表。

## G.5 内存映射

另一个虚拟内存项目是添加某种形式的内存映射文件。可能最简单的方法，是从可执行文件中惰性加载代码页。更全面的方法，是构建 mmap()系统调用和所有必要的基础结构，以便在页故障时从磁盘换入页面。

## G.6 内核线程

该项目探讨如何为 xv6 添加内核线程。clone()系统调用的操作与 fork 类似，但使用相同的地址空间。学生必须弄清楚如何实现这样的调用，以及如何创建真正的内核线程。学生还应该在其上构建一个小的线程库，提供简单的锁。

## G.7 高级内核线程

学生在其内核线程之上构建一个完整的线程库，添加不同类型的锁（自旋锁，在处理器不可用时休眠的锁）以及条件变量。还要添加必需的内核支持。

## G.8 基于范围的文件系统

第一个文件系统项目为基本文件系统添加了一些简单的功能。对于 EXTENT 类型的文件，学生将 inode 更改为存储范围（即指针—长度对）而不仅仅是指针。作为文件系统的相对简单的介绍。

## G.9 快速文件系统

学生将基本的 xv6 文件系统转换为 Berkeley 快速文件系统（FFS）。学生构建一个新的 mkfs 工具，引入块组和新的块分配策略，并构建大文件异常。在更深层次上理解文件系统工作原理的基础知识。

## G.10 日志文件系统

学生为 xv6 添加了一个基本的日志层。对于每次写入文件，日志 FS 会批量处理所有脏

块，并在磁盘日志中，为待写入的更新添加一条记录，然后再修改原来位置的块。通过引入崩溃点并展示文件系统始终恢复到一致状态，学生证明其系统的正确性。

## G.11 文件系统检查器

学生为 xv6 文件系统构建一个简单的文件系统检查程序。学生将了解文件系统的一致性，以及如何检查文件系统。