2025《操作系统》实验须知

March 7, 2025

2025

> ⚠️使用 Git 下载指定的 repo 完成实验
>
> Online Judge 将使用我们的脚本、Makefile，并在我们的环境下进行评测。因此如果你在本地修改了编译选项 (如去掉了 `-Wall -Werror` 等)、硬编码了路径 (例如 `"/home/jyy/log.txt"`) 等，提交后可能会发生编译/运行错误。
>
> 请大家自觉不要把自己的实验作业代码公开。如果你本着 "炫耀" 的态度公布，那你的代码很可能写得很烂不值得炫耀。请停止这种对学弟和学妹造成伤害的行为——如果你看到了互联网上的代码，请记得 Academic Integrity 且主动不要使用它们。

1. 获取实验框架代码

本课程所有实验都托管在同一个仓库中。与 Online Judge 评测环境一致。在命令行中运行

```
$ git clone https://git.nju.edu.cn/leo/os-workbench.git
```

获得框架代码，将会克隆 `os-workbench` 到当前目录。首次 clone 后你会得到一个近乎为空的 Git repo：

```
.
├── .git/
├── .shadow/
├── .gitignore
├── Makefile
└── oslabs.mk
```

每个实验的指南中都有获取该实验框架代码的说明。请妥善保管 os-workbench 目录：它保留了你完成作业的证据。如果在多个地点完成作业，请将整个目录移动 (或通过版本控制) 保持 Git 记录的完整。如遇问题请联系老师或助教。

关于本课程的实验环境，我们不做硬性要求，但我们推荐 Ubuntu 22.04，因为：

1. 作为使用较多的Linux系统，Ubuntu相关的环境问题的帖子比较多。当你们遇到环境问题时，比较容易在网上找到解答。
2. Online Judge的基础环境是Ubuntu 22.04。

2. 提交实验作业

我们已经为选修课程的同学生成了唯一的秘钥，请加入NJU Table的课程群组查看。如果有遗漏，请联系助教。为了提交你的代码至 Online Judge，你需要在 `oslabs.mk` 下面的 `???` 处填入你的token：

```
export TOKEN := ???
```

配置好 `TOKEN` 环境变量后，在相应的实验目录中 (例如， `M1` 的实验目录是 `pstree/`) 中执行以下命令完成提交：

```
$ make submit
```

如果提交成功，命令行中会看到：

```
$ make submit
[SUCC ✓] Received OS2025-M1 学号 (姓名) upload.tar.bz2 at 20:17:26
```

提交成功后，可以在评测网站上输入你的token以查看评测结果。注意我们只收取 `os-workbench/.git` 和目录中的 pdf 文件 (实验报告)。因此，**如果你只是修改了代码而没有执行过 make 或手工的 git commit，这些改动将不会被反映到 Online Judge**。

3. 使用 Git 管理源代码

在得到 Git repo 以后，默认处于 `main` 分支。你可以本学期全部在 `main` 分支上工作，但也可以自由创建自己的分支。

特别注意： `make` 会自动将你的实验代码保存到 `.shadow` 中 (为什么？)，这部分 Makefile 请不要修改。此外，请保留 Git 追踪部分，Git 记录将会作为我们筛选、检查提交的参考。如果你因为意外丢失了 Git 记录，只要你遵守学术诚信，就不必担心，Git 记录不参与评分。评分以 `.shadow` 中的代码为准。

4. 实验与评分

4.1 Mini Labs (`M1`, `M2`, ...)

1. 编写**可移植的代码**。我们会在 32/64-bit 两个平台上测试你的代码，因此请不要对指针类型的大小等作出假设。
2. **全程只有一个 C 源代码文件，请尽量控制在 500 行以内**。参考实现一般在 100-200 行，测试通过即得满分。
3. **只允许使用指定的 Makefile 编译 (使用 `make`)、只允许编辑已有的一个 `.c` 文件**。评测时，我们仅复制这一个 C 文件 (添加其他文件在 Online Judge 会导致编译错误)。
4. **不需要实验报告**。

4.2 OS Labs (`L1`, `L2`, ...)

1. 同样需要编写**可移植的代码**。我们将在 native, x86_64-qemu, x86-qemu 三个平台上测试。
2. **需要撰写实验报告** (以 pdf 格式存储在实验目录中，参考各个实验的要求)。除非特殊情况，实验报告不建议超过 2 页 A4 纸。请在实验报告中描述你在实验中遇到的特别值得一提的事件，例如你代码的架构设计、特别精巧的实现、遇到印象深刻的 bug 等。无需事无巨细交代清楚——好的代码不言自明。
3. 对于操作系统内核实验，后续实验的实验报告在前一次实验基础上追加。

4.3 评分规则

测试用例分为两个等级 (easy 和 hard)，easy 通常是一些 "冒烟测试" (smoke test)，即使用最典型简单的方式运行程序，检查程序是否 crash 以及输出合理的结果。Hard 则是更接近实际应用场景的测试用例。虽然你不能看到程序的日志输出 (否则测试用例很容易泄露)，但我们会对每个测试用例提供一定的解释，以帮助大家诊断问题。

5 Online Judge 环境

Mini/OS Lab 都在 Online Judge 评测。程序在容器中编译、运行，并由机器自动判定结果是否正确。具体环境为Ubuntu 22.04 容器 (Docker, x86-64)，容器中仅有最小的必要系统工具。你可以使用以下 Dockerfile 配置与在线评测一致的环境用于模拟Online Judge。

```
FROM ubuntu:22.04
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update
RUN apt-get install -y build-essential gcc-multilib qemu-system strace gdb sudo python3 libsdl2-dev
libreadline-dev
RUN apt-get upgrade -y
```

Mini Labs 直接在容器中执行 (non-root user)；OS Labs 在容器中的 QEMU 虚拟机 (tcg 模式) 运行；容器总内存限制 2GB，超过内存限会导致进程被杀死。超过一定时限未执行完的容器也将被杀死。容器中的编译器版本：

- gcc 11.4.0
- bin utils 2.38
- GNU make 4.3
- QEMU 6.2.0

Online Judge 的最大特点就是严格。有任何差错 (因为环境/配置等引发的编译错、细小的输出错误) 都将被 Online Judge 捕捉到。这有助于帮助大家摆脱 "糊弄" 的习惯，编写正确的程序。

本门课程的Labs与大家以往的程序设计课或者数据结构课的题目有一点不同：大部分问题没有 "绝对正确" 的标准输出。因此我们并不是简单地运行程序、比对结果，而是有一定系统化地测试你的程序：

- 在多个环境下运行你的程序，如 i386 (32 位) 和 x86-64 (64位)，因此不可移植的代码可能无法编译；
- 在模拟出的环境中执行程序，例如在线程调度时插入一些随机的 delays，从而提高某些并发 bug 触发的概率；
- 链接我们修改过的库函数，例如 (在某些 lab 中) 使 `malloc()` 随机返回 `NULL`；
- 解析程序的 log，并观察其中是否有 bug 出现的迹象。例如程序 crash 将被判定为不正确、缺少某个重要输出也将被判定为不正确。