操作系统实验

lab0　初识操作系统

内容提要

- 实验课程的整体说明
- 背景知识
- 实验概述
- 实验内容
  - 掌握实验相关工具的使用
  - 掌握git的管理模式
  - 实验完成方式
- 实验实战
- 实验提交

整体说明

实验设置：

实验	实验开始时间（讲解实验）	第一次课上考察时间	第二次课上考察时间	实验报告deadline
Lab 0：Linux、Git 基础知识	第2周	第3周	--	第4周周一23:59
Lab 1：内核制作、启动和 printf	第3周	第4周	第5周	第6周周一23:59
Lab 2：内存管理	第5周	第6周	第7周	第8周周一23:59
Lab 3：进程与异常	第7周	第8周	第9周	第10周周一23:59
Lab 4：系统调用与 fork	第9周	第10周	第11周	第12周周一23:59
Lab 5：文件系统	第11周	第12周	第13周	第14周周一23:59
Lab 6：管道和命令解释程序	第13周	--	--	第16周周一23:59

整体说明

- 操作系统是计算机系统的一个重要系统软件，本课程通过实验，希望学生了解实际操作系统的工作过程，在实践中加深对操作系统原理的理解。
- 本课程配合计算机操作系统原理课，加强操作系统的实验环节，让计算机专业本科学生综合地运用所学专业知识解决操作系统的分析和设计问题，了解操作系统的开发过程以及具体算法的实现环节。
- 课程设计MOOC网址：
  http://cscore.net.cn/courses/course-v1:Internal+B3I062140+2020_T2/info
- 课程设计实验操作网址：
  https://buaaos.studio

整体说明

实验要求：

- **课下测试：在线独立完成每个 OS实验编程任务（Lab0-Lab6，共7个实验），在实验环境中提交并通过自动评测测试。**
- **课上测试：在实验课排课时间到机房完成现场发布的题目，在实验环境中提交并通过自动评测测试。本学期共安排11次课上测试，其中Lab0安排1次，Lab1~Lab5各安排2次。每次课上测试包括一道exam题目和一道extra题目，要求在规定的时间内完成。**

整体说明

实验要求：

- **及格条件：通过至少5次课上测试的exam题目。如果通过课上测试的exam题目少于5次，无论基础分与加分有多少，最终操作系统课程设计部分成绩不超过59分。**
- **申优条件：完成Lab0-Lab6所有课下测试（必须全部都在deadline前完成）并按时提交实验报告，且完成所有课上测试exam题目和至少两次课上测试的extra题目，且完成任意一项挑战性任务。**
- **实验作业：完成课下测试后在课程中心提交实验报告**

背景知识

- 工欲善其事必先利其器
- 操作系统课程设计需要同学们大量使用Linux操作系统的命令行交互界面，这意味着我们需要掌握一些基础操作命令，同时需要硬件模拟器GXemul模拟mips运行环境，最终由Git进行版本控制和评测。
- 课程服务器会为同学们提供成熟的实验环境，但考虑到很多同学对于没有图形化的操作系统界面几乎没有接触，故本章会着重为同学们进行实验环境的介绍，旨在“扫盲”
- 熟练掌握本章内容会让同学们后续实验如鱼得水

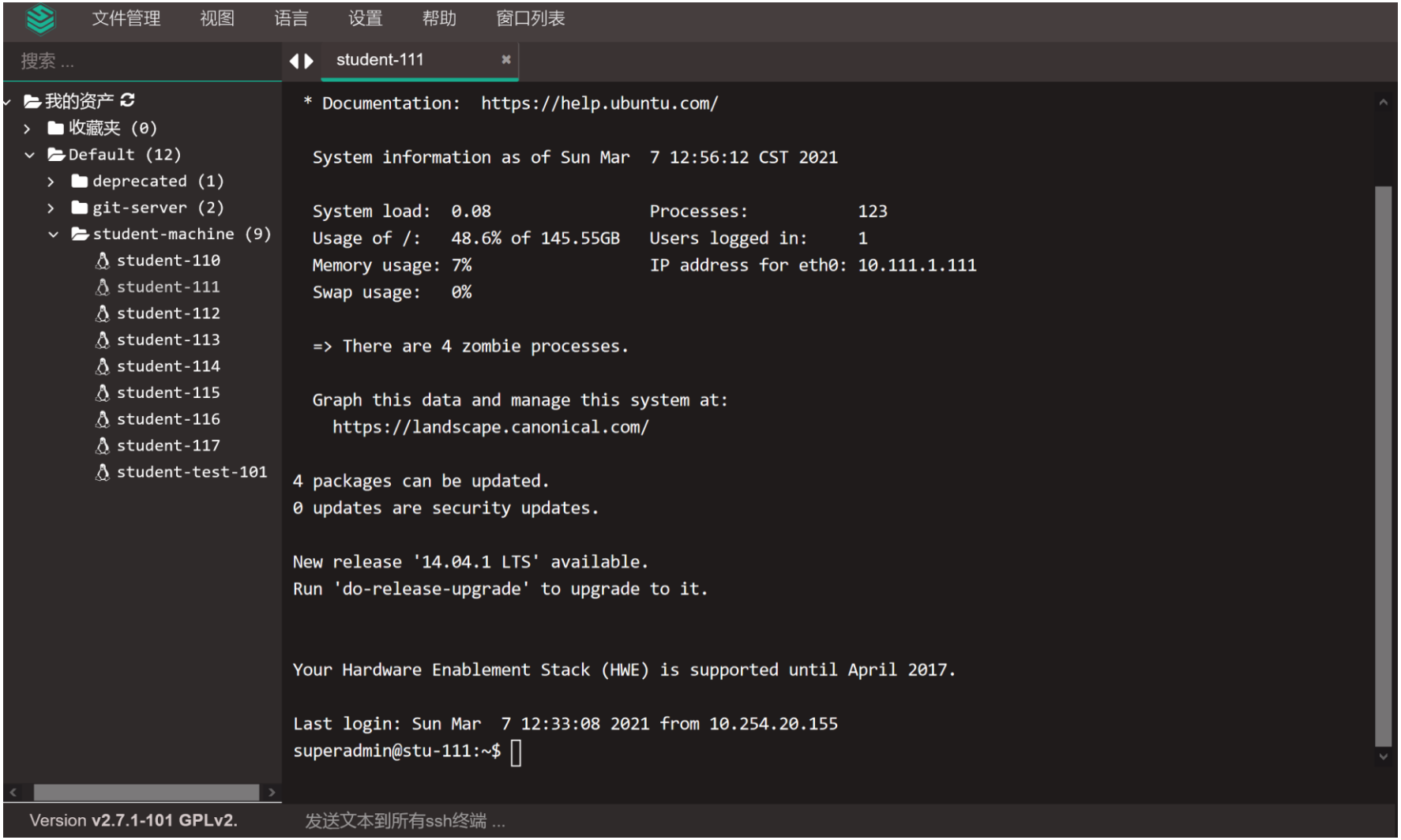
实验概述

- 熟悉实验环境配置
- 连接工具：远程连接到我们的实验环境
- 操作系统：Linux 虚拟机Ubuntu，需掌握常用命令
- 硬件模拟器：GXemul，本章暂不需要了解
- 编译器：GCC，需掌握命令格式及常用选项
- 版本控制：Git，需掌握如何提交本地变更，并同步到远端库以触发评测脚本；了解版本回退等实用功能
- Makefile
- 掌握相关技巧后，完成课下测试任务，提交评测

实验内容—实验平台

- 具体实验所需环境已经集成到实验平台上(https://buaaos.studio)，每位同学都有自己的账号和密码，登陆即可。
- 教学系统网站可以直接打开Web终端，点击左端的 Web终端 即可进入终端操作页面（具体操作查看实验平台介绍PPT)

实验内容—实验平台


文件管理
视图
语言
设置
帮助
窗口列表
搜索 ...
student-111
我的资产
收藏夹 (0)
Default (12)
deprecated (1)
git-server (2)
student-machine (9)
student-110
student-111
student-112
student-113
student-114
student-115
student-116
student-117
student-test-101
* Documentation: https://help.ubuntu.com/
System information as of Sun Mar 7 12:56:12 CST 2021
System load: 0.08
Usage of /: 48.6% of 145.55GB
Memory usage: 7%
Swap usage: 0%
Processes: 123
Users logged in: 1
IP address for eth0: 10.111.1.111
=> There are 4 zombie processes.
Graph this data and manage this system at:
https://landscape.canonical.com/
4 packages can be updated.
0 updates are security updates.
New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Your Hardware Enablement Stack (HWE) is supported until April 2017.
Last login: Sun Mar 7 12:33:08 2021 from 10.254.20.155
superadmin@stu-111:~$
Version v2.7.1-101 GPLv2.
发送文本到所有ssh终端 ...

实验内容—获取lab0环境

- git clone git@10.210.0.67:yourid_2021-lab

其中yourid改为自己的学号

- git branch -a（查看远程仓库分支）
- git checkout lab0　　（此时成功将lab0环境拷贝到当前目录，即可进行实验）
- git pull　（拉取lab0分支）

实验内容--基础操作命令

- ls：显示当前目录文件
- mkdir：建立目录文件
- rmdir：删除空目录
- rm：删除文件
- cd：变更工作目录
- cat：连接并输出文件
- cp：复制文件
- mv：移动文件
- source：运行可执行文件
- find：查找文件
- grep：文本搜索，可跨文件
- man：帮助手册

实验内容—实用工具

- nano 的主界面如下图所示，所有基本的操作 都被罗列在下面。nano 较为容易上手，但功能相对有限。
- Vim 被誉为编辑器之神，是程序员为程序员设计的编辑器，编辑效率高，十分适合 编辑代码，其界面如下图所示。

实验内容—实用工具


GNU nano 2.2.6 文件： Test
这是一个使用nano编辑器打开的文本文件
可以看到屏幕下方有常用基本操作的按键提示
其中界面上显示的向上的尖状符号（^）代表 Ctrl，后面的字母就代表相应的快捷键
例如： Ctrl+G 打开辅助说明
Ctrl+O 保存文件
Ctrl+W 搜索
Ctrl+X 退出nano
...
更多的按键操作说明可以使用 Ctrl+G 打开辅助说明查看
^G 求助
^X 离开
^O 写入
^J 对齐
^R 读档
^W 搜索
^Y 上页
^V 下页
^K 剪切文字
^U 还原剪切
^C 游标位置
^T 拼写检查

实验内容—实用工具

```
1  ================================================================================
2  这是一个使用Vim编辑器打开的文本文件
3  刚打开Vim时会默认进入命令模式
4  在命令模式中，可以使用Vim的操作命令对文本进行操作
5  只有在插入模式中，才可以键入字符
6  下面列举一些Vim的常用基本操作：
7          (在命令模式下)
8          i        切换为插入模式
9          Esc      返回普通模式
10         o        在当前行之下插入
11         O        在当前行之上插入
12         u        撤销(undo)
13         Ctrl+r   重做(redo)
14         yy       复制一行
15         dd       剪切一行
16         y        复制(按y后按方向键左/右,复制光标左/右边的字符)
17         d        剪切(按d后按方向键左/右,剪切光标左/右边的字符)
18         2yy      复制下面2行
19         3dd      剪切下面3行
20         4y       复制4个字符(按方向键左/右开始复制光标左/右的字符)
21         5d       剪切5个字符(按方向键左/右开始剪切光标左/右的字符)
22         p        在当前位置之后粘贴
23         P        在当前位置之前粘贴
24         :q       不保存直接退出
25         :q!      强制不保存退出
26         :w       保存
27         :wq      保存后退出
28         :N       将光标移至第N行
29         :set nu  显示行号
30         /word    查询word在文中出现的位置，若有多个，按n/N分别移至下/上一个
31 ================================================================================
                                                              31,80         全部
```

实验内容—实用工具2

- GCC
  - 在没有 IDE 的情况下，使用 GCC 编译器是一种简单快捷生成可执行文件的途径， 只需一行命令即可将 C 源文件编译成可执行文件。

实验内容—实用工具2

```
语法:gcc [选项]... [参数]...
选项（常用）:
                -o                  指定生成的输出文件
                -S                  将 C 代码转换为汇编代码
                -wall           显示警告信息
                -c                  仅执行编译操作，不进行链接操作
                -M                  列出依赖
参数：
        C 源文件：指定 C 语言源代码文件
e.g.

$ gcc test.c
# 默认生成名为 a.out 的可执行文件
#Windows 平台为 a.exe

$ gcc test.c -o test
# 使用-o 选项生成名为 test 的可执行文件
#Windows 平台为 test.exe
```

实验内容--Git介绍

- 最原始的版本控制是纯手工的版本控制：修改文件，保存文件副本。有时候偷懒省事，保存副本时命名比较随意，时间长了就不知道哪个是新的，哪个是老的了，即使知道新旧，可能也不知道每个版本是什么内容，相对上一版作了什么修改了，当几个版本过去后，很可能就是下面的样子了：

实验内容--Git介绍

- Git则是一个：
  - 1.自动记录每次文件的改动，可以轻松撤销改动。
  - 2.多人作编辑不费力，有着简洁的指令与操作。
  - 3.可以像时光机一样穿越回以前，而且不但能穿越回去，还能在不满意的时候穿越回来！
  - 4.如果想查看某次改动，只需要在软件里瞄一眼就可以。

实验内容--Git介绍

- git init: 初始化当前目录为Git工作区
- git add: 追踪文件变更
- git commit: 提交文件变更
- git log: 查看提交记录
- git status: 查看文件状态
- git reset: 版本回退
- git branch <branch-name>: 创建新分支
- git checkout <branch-name>: 切换分支
- git push: 将本地变更推送到远端仓库
- git pull: 将远端仓库抓回到本地库

Makefile

- target: dependencies
  - command 1
  - command 2
  - ...
  - command n

- all: hello_world.c
  - gcc -o hello_world hello_world.c

- http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor
- http://www.gnu.org/software/make/manual/make.html#Reading-Makefiles

实验实战

- 1、在lab0工作区的src目录中，存在一个名为palindrome.c的文件，使用刚刚学过的工具打开palindrome.c，使用c语言实现判断输入整数n(1≤n≤10000)是否为回文数的程序(输入输出部分已经完成)。通过stdin每次只输入一个整数n，若这个数字为回文数则输出Y并退出程序，否则输出N并退出程序。**[注意：正读倒读相同的整数叫回文数]**
- 2、在src目录下，存在一个未补全的Makefile文件，借助刚刚掌握的Makefile知识，将其补全，以实现通过make指令触发src目录下的palindrome.c文件的编译链接的功能，生成的可执行文件命名为palindrome。

实验实战

3、在src/sh_test目录下，有一个file文件和hello_os.sh文件。

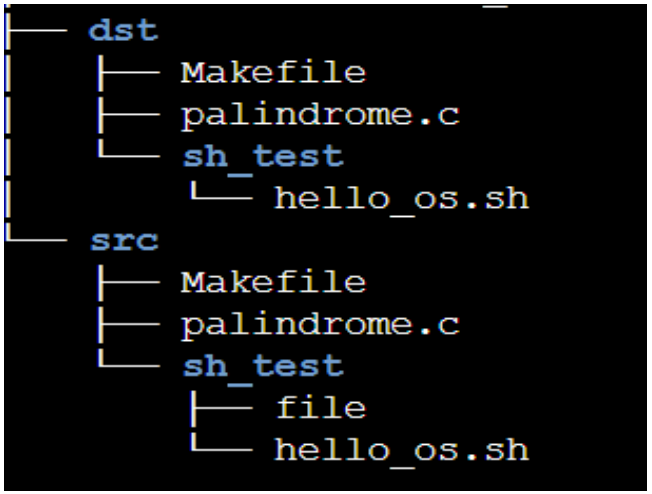
hello_os.sh是一个未完成的脚本文档，请同学们借助shell编程的知识，将其补完，以实现通过指令bash hello_os.sh AAA BBB，在hello_os.sh所处的文件夹新建一个名为BBB.c的文件，其内容为AAA文件的第8、32、128、512、1024行的内容提取(AAA文件行数一定超过1024行)。**[注意：对于指令bash hello_os.sh AAA BBB，AAA及BBB可为任何合法文件的名称，例如 bash hello_os.sh file hello_os.c，若已有hello_os.c文件，则将其原有内容覆盖]**

- 4、补全后的palindrome.c、Makefile、hello_os.sh依次复制到路径 /dst/palindrome.c /dst/Makefile /

实验实战

完成Step1~Step4:

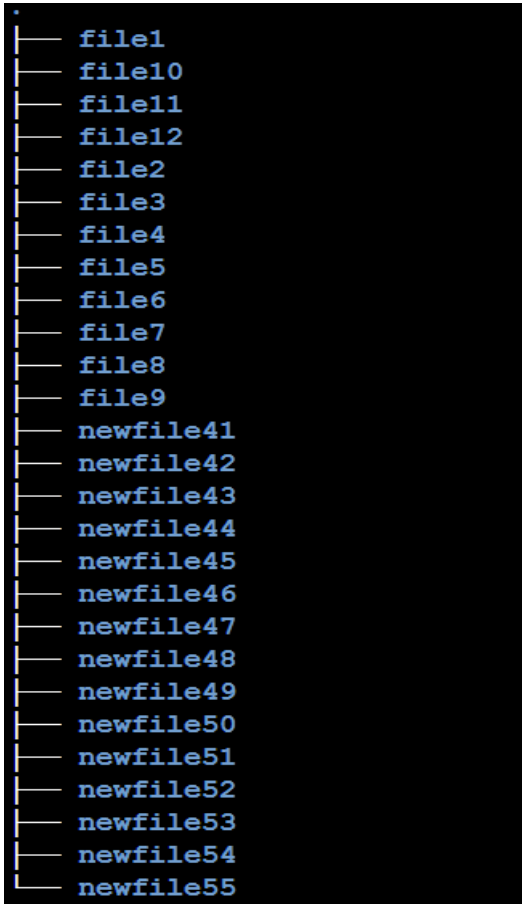
要求按照测试1~测试4要求完成后，最终提交的文件树图示如下：



实验实战

5、在lab0工作区ray/sh_test1目录中，含有100个子文件夹file1~file100，还存在一个名为changefile.sh的文件，将其补完，以实现通过指令bash changefile.sh，可以删除该文件夹内file71~file100共计30个子文件夹，将file41~file70共计30个子文件夹重命名为newfile41~newfile70，保留file1~file40共计40个子文件夹。[**注意：评测时仅检测changefile.sh的正确性**]

实验实战



完成Step5：

要求按照测试5要求完成后，最终提交的文件树图示如左(图中只显示了部分文件夹，file下标只显示1~12，newfile下标只显示41~55)

实验实战

- 6、在lab0工作区的ray/sh_test2目录下，存在search.sh文件，将其补完，以实现通过指令bash search.sh file int result，可以在当前文件夹下生成result文件，内容为file文件所有含有int字符串对应的行数，即若有多行含有int字符串需要全部输出。**[注意：对于指令bash search.sh file int result，file及result可为任何合法文件名称，int可为任何合法字符串，若已有result文件，则将其原有内容覆盖，匹配时大小写不忽略]**

实验实战

完成Step6:

要求按照测试6要求完成后，result内显示样式如下(一个答案占一行):

实验实战

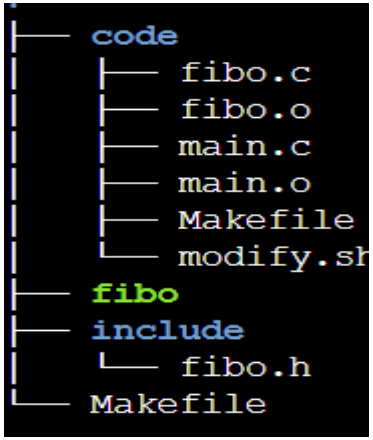
- 7、在lab0工作区的csc/code目录下，存在fibo.c、main.c，其中fibo.c有点小问题，还有一个未补全的modify.sh文件，将其补完，以实现通过指令bash modify.sh fibo.c char int，可以将fibo.c中所有的char字符串更改为int字符串。**[注意：对于指令bash modify.sh fibo.c char int，fibo.c可为任何合法文件名，char及int可以是任何字符串，评测时评测modify.sh的正确性，而不是检查修改后fibo.c的正确性]**
- 8、lab0工作区的csc/code/fibo.c成功更换字段后(bash modify.sh fibo.c char int)，现已有csc/Makefile和csc/code/Makefile，补全两个Makefile文件，要求在csc目录下通过指令make可在csc/code文件夹中生成fibo.o、main.o，在csc文件夹中生成可执行文件fibo，再输入指令make clean后只删除两个.o文件。**[注意：不能修改fibo.h和main.c文件中的内容，提交的文件中fibo.c必须是修改后正确的fibo.c，**

实验实战

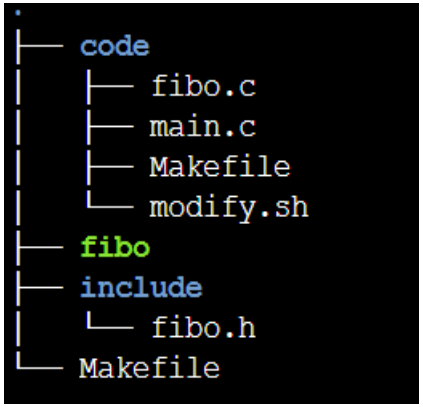
完成Step7~8:

要求成功使用脚本文件modify.sh修改fibo.c，实现使用make指令可以生成.o文件和可执行文件，再使用指令 make clean可以将.o文件删除，但保留fibo和.c文件。（最终提交时文件中fibo和.o文件可有可无）

make后文件树



make clean后文件树



实验提交

- modify 写代码。
- git add --all
- git commit <modified-file> 提交到本地版本库。
- git pull 从服务器拉回本地版本库，并解决服务器版本库与本地代码的冲突。
- git add
- git commit <conflict-file> 将远程库与本地代码合并结果提交到本 地版本库。
- git push 将本地版本库推到服务器

实验提交

- 本地实验完成后，初次提交进行评测输入git commit -m "xxx"后，还需要git config --global设置好个人的用户名和邮件，后面再提交不需要设置。初次提交如下

```
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git add --all
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git commit -m "xxx"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'jovyan@94e1415dbf93.(none)')
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git config --global user.email "you@example.com"
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git config --global user.name "Your Name"
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git commit -m "xxx"
[lab0 ddc80e7] xxx
 1 file changed, 2 insertions(+)
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git push origin lab0:lab0
```

-

```
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git add --all
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git commit -m "xxx"
[lab0 aa12545] xxx
 1 file changed, 1 insertion(+)
jovyan@94e1415dbf93:~/work/JU4K32RIIUEKC-lab$ git push origin lab0:lab0
```

实验提交

```
remote: make --directory=code
remote: make[1]: Entering directory '/usr/src/workdir/code'
remote: gcc -c fibo.c -I./../include
remote: gcc -c main.c -I./../include
remote: make[1]: Leaving directory '/usr/src/workdir/code'
remote: gcc code/fibo.o code/main.o -o fibo
remote: [ fibo found. ]
remote: [ You have passed fibo testcase 1/2 ]
remote: [ You have passed fibo testcase 2/2 ]
remote: [ find your fibo.o ]
remote: [ find your main.o ]
remote: make --directory=code clean
remote: make[1]: Entering directory '/usr/src/workdir/code'
remote: rm *.o
remote: make[1]: Leaving directory '/usr/src/workdir/code'
remote: [ make clean can delete fibo.o ]
remote: [ make clean can delete main.o ]
remote: [ find your fibo after make clean ]
remote: [ You got 100 (of 100) this time. 2021年 03月 08日 星期一 08:41:02 CST ]
```

实验提交

成功提交分数≥60，在下一个lab课下实验开放后，需要重新提交触发评测后获取下一个代码分支，流程如下：

- 修改任意文件（增加或删除空行）
- `git add .`
- `git commit`
- `git push` 如果当前分支的暂存区还有东西的话，先提交，没有可忽略。
- `git pull` 这一步很重要！要先确保服务器上的更新全部同步到本地版本库！
- `git checkout lab1` 检出新实验分支并进行实验。（注：获取lab1分支后应该重新进入对应实验平台进行实验）