

P01: UNIX V6++ 运行调试环境的安装与配置实验报告

姓名：范潇 学号：2254298 日期：2024 年 9 月 5 日

1. 题目一

1.1. 实验目的

安装配置 UNIX V6++ 的运行环境。

1.2. 实验内容

1. 安装后端服务根证书；
2. 登录实验平台；
3. 运行远程桌面环境；
4. 初始化代码仓库；
5. 启动 UNIX V6++ 运行环境。

1.3. 实验过程

安装完后端服务根证书后，登录网站 <http://vesper-system.pages.tongji.edu.cn/vesper-front/>，然后如图1所示，修改密码。接着，如图 2所示，修改主机名。之后，我在 GitLab 中创建了自己的代码分支。等待桌面环境启用后，如图3所示，生成密钥，然后如图4所示，将公钥上传至 GitLab。当我尝试下载实验工具包时，要求我输入用户名和密码，为此，为如图5所示，申请了个人访问令牌用作密码。下载完成后，将工作目录设置为 `unix-v6pp-tongji`，然后如图7,8,9所示，依次执行 `init.sh` 和 `make qemu` 命令，最终启动了 UNIX V6++ 界面并如图10所示，执行了 `echo` 命令。

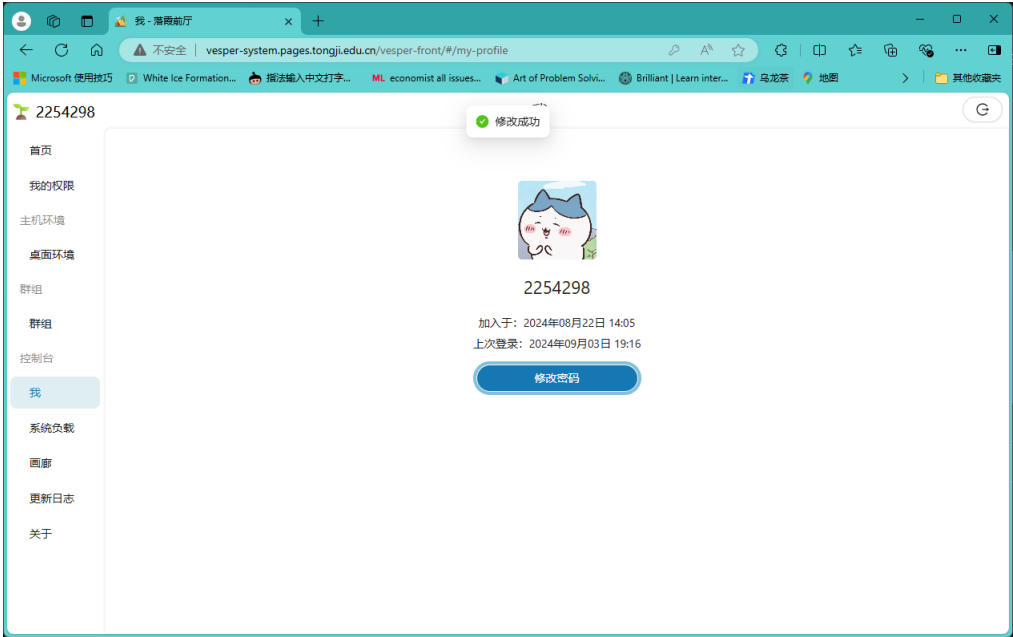


图 1: 修改密码

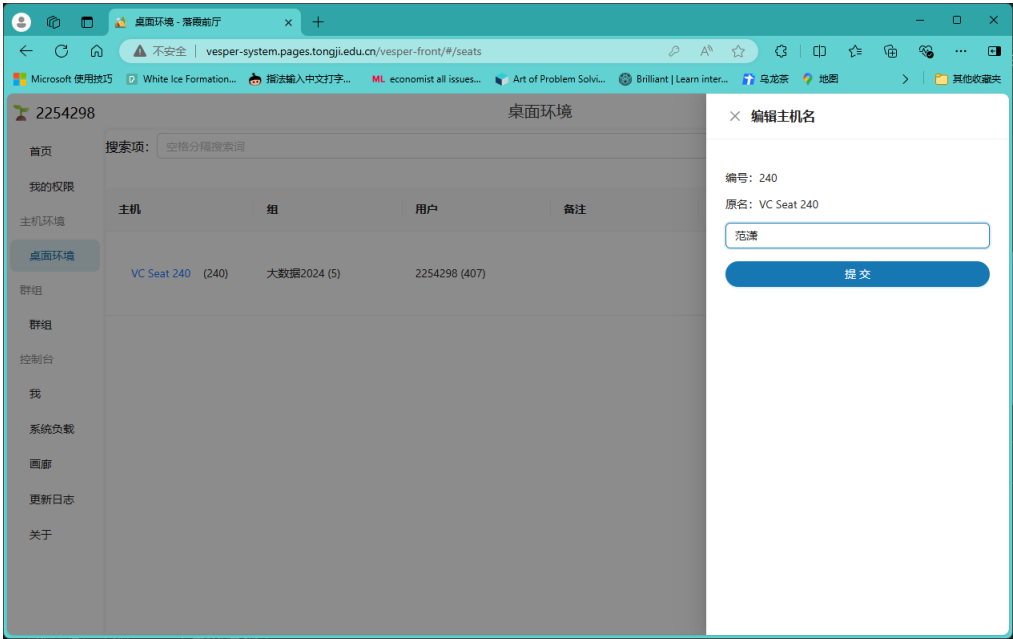


图 2: 修改主机名

```
[vesper_center_240@archlinux .ssh]$ ssh-keygen -t ed25519 -C "2254298@tongji.edu.cn"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/vesper_center_240/.ssh/id_ed25519):
/home/vesper_center_240/.ssh/id_ed25519 already exists.
Overwrite (y/n)?
[vesper_center_240@archlinux .ssh]$ cat id_ed25519.pub
```

图 3: 生成密钥



图 4: 在 GitLab 中上传公钥

添加一个个人访问令牌

输入您的应用程序的名称，我们会返回唯一的个人访问令牌。

令牌名称

AnikiFan

例如，使用令牌的应用程序或令牌的用途。不要提供令牌名称的敏感信息，因为它将对所有 project 成员可见。

到期时间

2025-03-01

选择范围

范围设置授予令牌的权限级别。 [了解更多。](#)

☒

api

授予对 API 的完全读/写访问权，包括所有群组和项目、容器镜像库和软件包库。

☒

read_api

授予对 API 的读访问权，包括所有群组和项目、容器镜像库和软件包库。

☒

通过 /user API 端点授予对通过身份验证的用户概要的只读访问权，该端点包括用户名、公共电子邮件和全名。还授予对 /users 下的只读 API 端点的访问权。

☒

使用 Git-over-HTTP 或 Repository Files API 授予对私有项目仓库的只读访问权。

☒

使用 Git-over-HTTP (不使用 API) 授予对私有项目上的仓库的读写访问权。

☒

授予对私有项目上的容器镜像库镜像的只读访问权。

☒

授予对私有项目上的容器镜像库镜像的写访问权。

创建个人访问令牌

图 5: 设置个人访问令牌

```
[vesper_center_240@archlinux ~]$ git clone https://git.tongji.edu.cn/2254298/unix-v6pp-tongji.git
Cloning into 'unix-v6pp-tongji'...
Username for 'https://git.tongji.edu.cn': 2254298
Password for 'https://2254298@git.tongji.edu.cn':
remote: Enumerating objects: 564, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 564 (delta 0), reused 3 (delta 0), pack-reused 561
Receiving objects: 100% (564/564), 1.82 MiB | 6.02 MiB/s, done.
Resolving deltas: 100% (236/236), done.
```

图 6: 下载实验工具包

```
[vesper_center_240@archlinux unix-v6pp-tongji]$ bash init.sh
Cloning into 'tools/unix-v6pp-filesystem-editor/src'...
```

图 7: 执行 init.sh

```
build success (fsedit & filescanner).
[vesper_center_240@archlinux unix-v6pp-tongji]$ make qemu
mkdir -p target/objs/asm-dump
```

图 8: 执行 make qemu

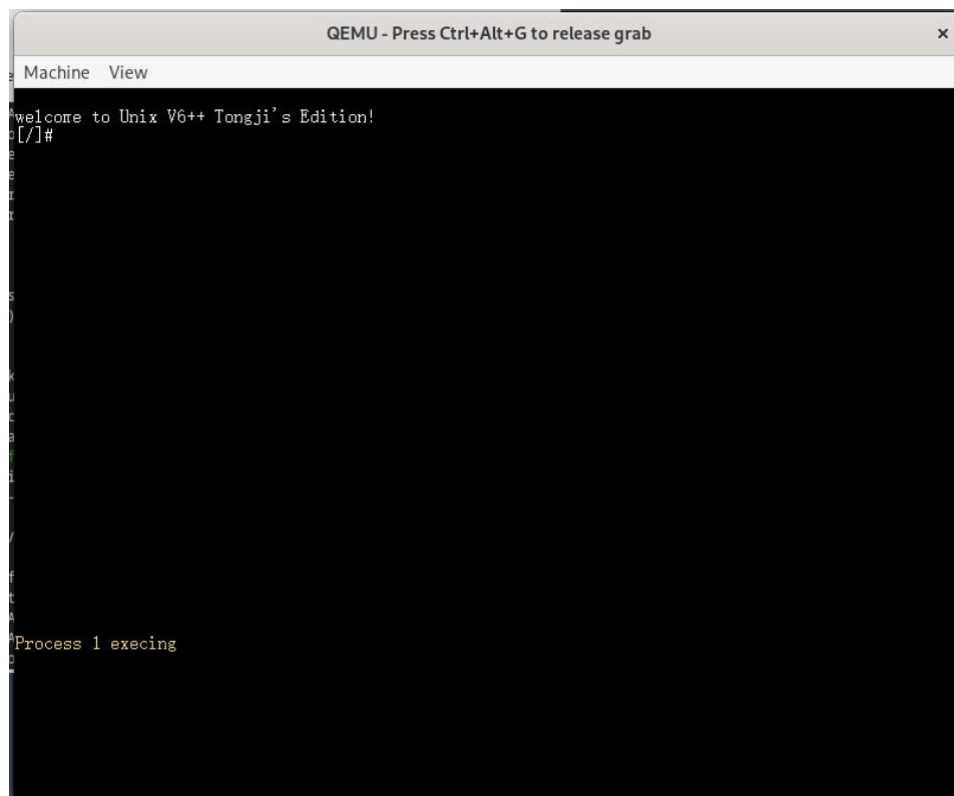
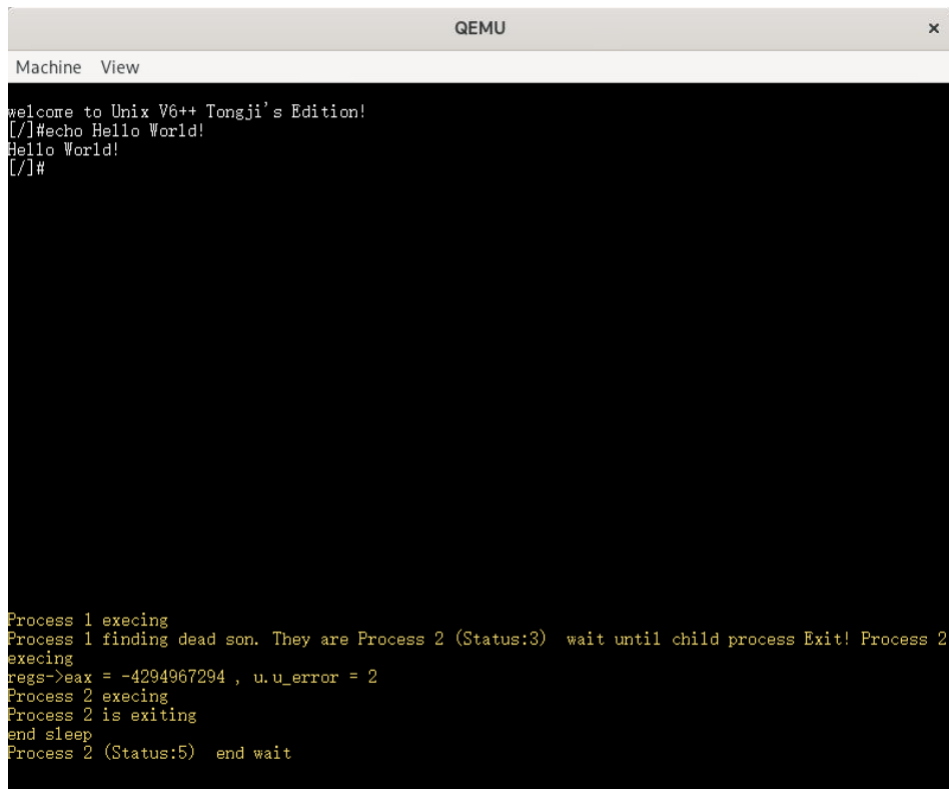


图 9: UNIX V6++ 界面



```
Machine View
welcome to Unix V6++ Tongji's Edition!
[/]#echo Hello World!
Hello World!
[/]#

Process 1 execing
Process 1 finding dead son. They are Process 2 (Status:3) wait until child process Exit! Process 2
execing
regs->eax = -4294967294 , u.u_error = 2
Process 2 execing
Process 2 is exiting
end sleep
Process 2 (Status:5) end wait
```

图 10: 执行 echo 命令

2. 题目二

2.1. 实验目的

安装配置 UNIX V6++ 的调试环境。

2.2. 实验内容

1. 启动 UNIX V6++ 的调试模式;
2. 利用本地 VSCODE 进行远程调试。

2.3. 实验过程

由于无法正常启动远程桌面环境中的 Vscode, 所以利用本地 Vscode 进行调试。首先在远程界面环境中执行 make qemug 命令, 然后再本地 Vscode 中连接远程桌面环境, 并打开 unix-v6pp-tongji 文件夹。接着将调试对象设置为内核, 并在 next 函数中设置断点。在 Vscode 中开始调试并命中断点后, 进行单步调试, 直至内核初始化完成。

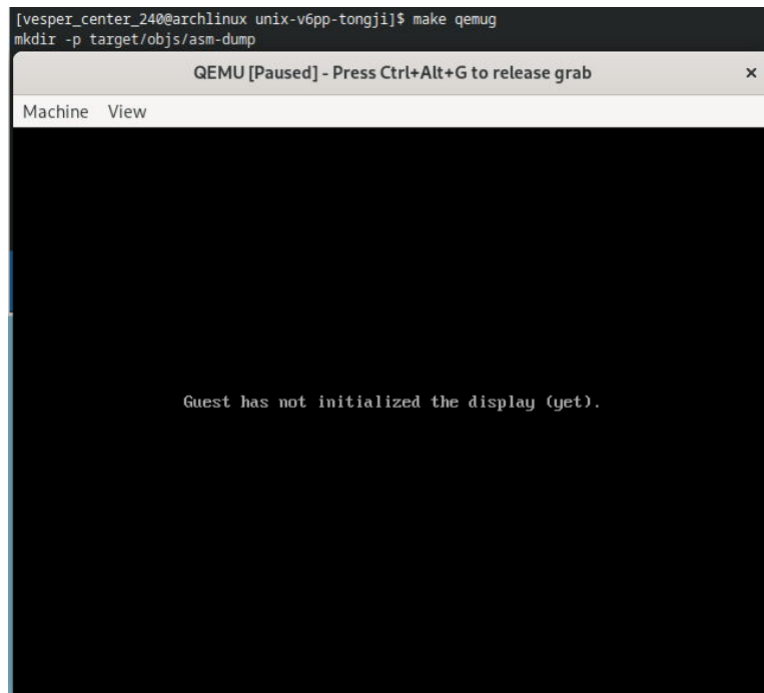


图 11: 执行 make qemu 命令

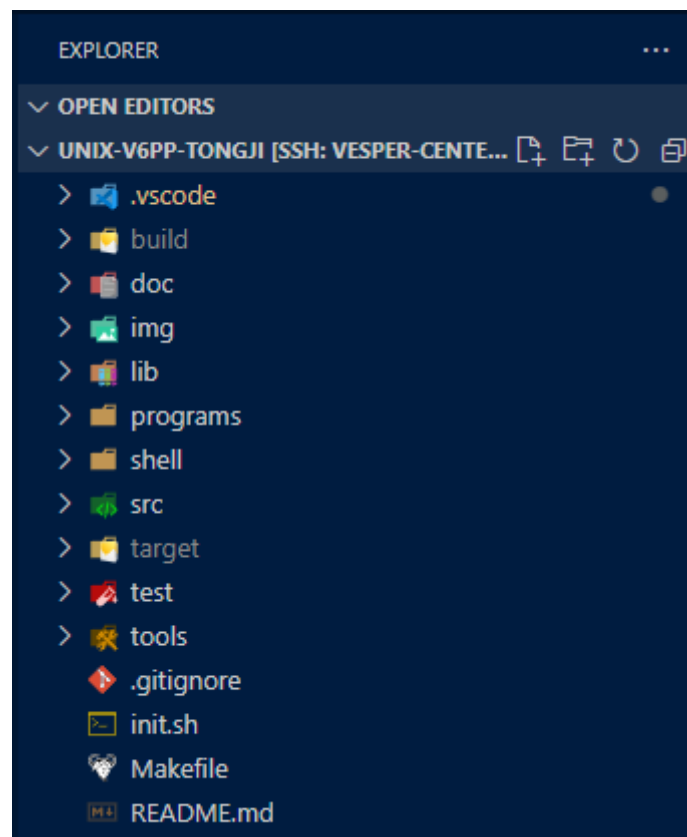


图 12: 打开 unix-v6pp-tongji 文件夹

```
8      "program": "${workspaceFolder}/target/objs/kernel.exe",
9      // "program": "${workspaceFolder}/target/objs/apps-elf/ls",
10     //"program": "${workspaceFolder}/target/objs/Shell.elf.exe",
```

图 13: 设置调试对象

```
206 extern "C" void next()
207 {
208
209     #ifdef USE_VESA
210         intptr_t vesaModeInfoAddr = Machine::KERNEL_SPACE_START_ADDRESS + 0x7e00;
211         auto& vesaModeInfo = * (video::svga::VbeModeInfo*) vesaModeInfoAddr;
212         video::svga::init(&vesaModeInfo);
213
214         Machine::Instance().InitVESAMemoryMap(
215             vesaModeInfo.framebuffer,
216             video::svga::VESA_SCREEN_VADDR,
217             video::svga::bytesPerPixel * vesaModeInfo.height * vesaModeInfo.width
218         );
219
220         video::console::init();
221         video::console::writeOutput("VESA enabled.\n", -1, 0xfeba07);
222
223     #endif
224 }
```

图 14: 设置断点

```
209  ▾ #ifdef USE_VESA
210      intptr_t vesaModeInfoAddr = Machine::KERNEL_SPACE_START_ADDRESS + 0x7e00;
211      auto& vesaModeInfo = * (video::svga::VbeModeInfo*) vesaModeInfoAddr;
212      video::svga::init(&vesaModeInfo);
213  
```

图 15: 命中断点

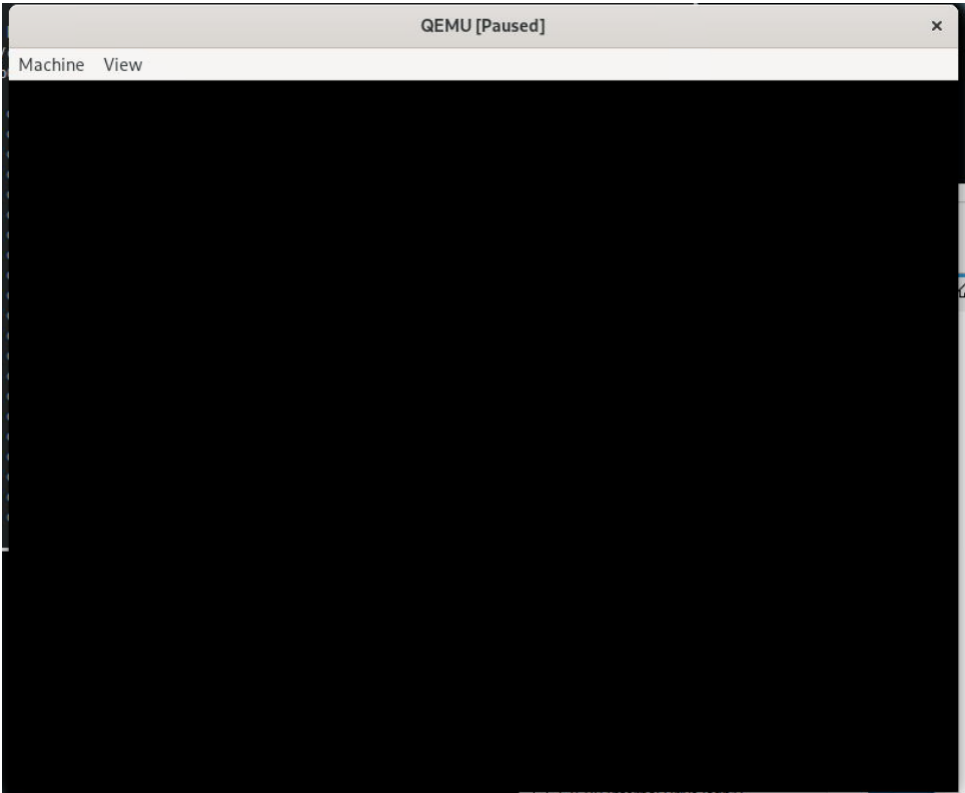


图 16: 命中断点时的界面

```
254     Kernel::Instance().Initialize();
255     Kernel::Instance().GetProcessManager().SetupProcessZero();
256     isInit = true;
```

图 17: 单步调试

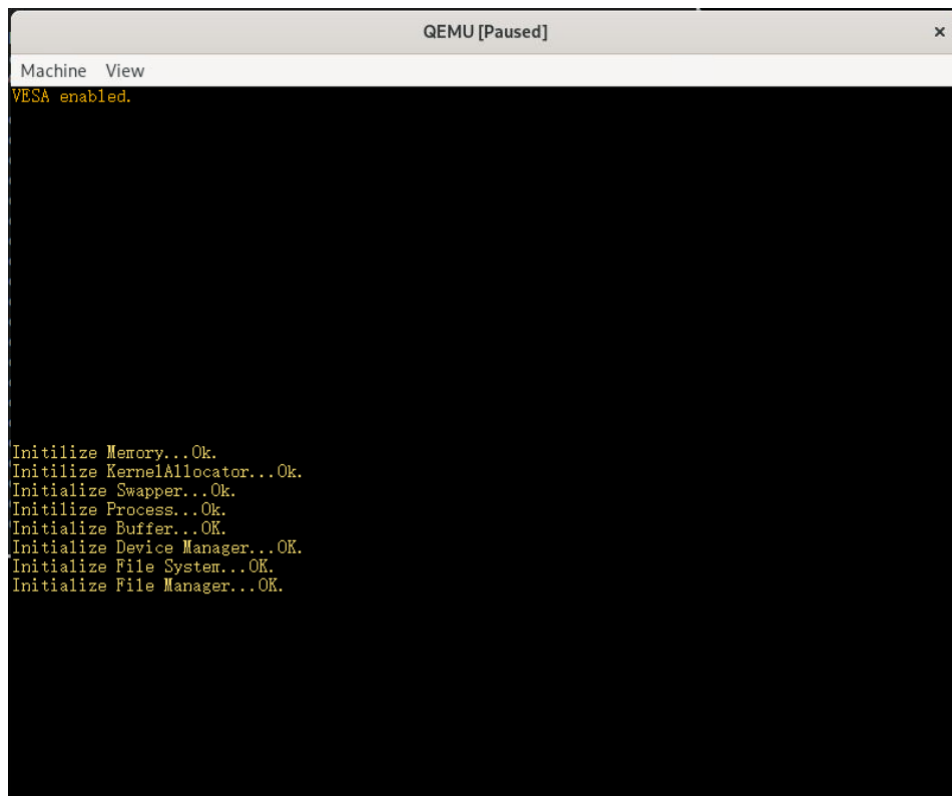


图 18: 内核初始化完成时的界面