OS 第二次作业

于新雨 计25 2022010841

1

配置环境总体顺利，未遇到问题

2

执行了 copy.c 编译出来的二进制文件，结果如下：

```
❯ strace ./copy test rosa yxy
execve("./copy", ["./copy", "test", "rosa", "yxy"], 0x7fffb161e7e8 /* 37 vars */)
= 0
brk(NULL)                               = 0x55cfd213d000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffcee475a0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f8f960a1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=50999, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 50999, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8f96094000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832)
= 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,
784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"...,
48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\315A\vq\17\17\tLh2\355\331Y1\0m"...,
68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,
784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f8f95e6b000
mprotect(0x7f8f95e93000, 2023424, PROT_NONE) = 0
mmap(0x7f8f95e93000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f8f95e93000
mmap(0x7f8f96028000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f8f96028000
mmap(0x7f8f96081000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f8f96081000
mmap(0x7f8f96087000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8f96087000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f8f95e68000
arch_prctl(ARCH_SET_FS, 0x7f8f95e68740) = 0
```

```
set_tid_address(0x7f8f95e68a10)         = 31837
set_robust_list(0x7f8f95e68a20, 24)     = 0
rseq(0x7f8f95e690e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f8f96081000, 16384, PROT_READ) = 0
mprotect(0x55cf9ea67000, 4096, PROT_READ) = 0
mprotect(0x7f8f960db000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0
munmap(0x7f8f96094000, 50999)           = 0
write(1, "test", 4test)                     = 4
write(1, " ", 1 )                          = 1
write(1, "rosa", 4rosa)                     = 4
write(1, " ", 1 )                          = 1
write(1, "yxy", 3yxy)                       = 3
write(1, "\n", 1
)                       = 1
exit_group(0)                           = ?
+++ exited with 0 +++
```

可以分为以下几个过程：

- 启动：调用了 execve，brk(初始化堆空间)，一系列 syscall 来加载 linker，一系列 syscall 来加载运行时库（libc.so.6），设置各段的 rwx 权限
- 执行：调用 write syscall，来依次打印命令行参数
- 结束：exit_group(0)

3

- man 是 Unix/Linux 系统中的命令行工具，用于查阅系统手册页，基本使用方式是在终端输入 man 后跟命令名称（如 man ls）来获取该命令的详细说明和用法
- 选择 prctl syscall，第一个参数是 code，指定执行操作的类型（对应具体功能的 subfunctions），往 subfunction 传递第二个 addr 参数，用来设置或者获取 value(关于某个进程或者线程的状态)
- 选择 ioctl syscall，这个函数的接受的参数数量和类型是不定的（就像是 printf 一样），第一个参数是 device 的 fd,第二个参数是 request code(ioctlcode)，指定功能的类型，后面的参数由驱动等自行决定

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <termios.h>
#include <sys/ioctl.h>

int main() {
    int fd;
    struct winsize ws;

    // Open the current terminal
    fd = open("/dev/tty", O_RDWR);
    if (fd < 0) {
```

```
        perror("Failed to open terminal");
        return 1;
    }

    // Use ioctl to get terminal window size
    if (ioctl(fd, TIOCGWINSZ, &ws) < 0) {
        perror("ioctl failed to get window size");
        close(fd);
        return 1;
    }

    printf("Terminal dimensions:\n");
    printf("Rows: %d\n", ws.ws_row);
    printf("Columns: %d\n", ws.ws_col);
    printf("Pixel width: %d\n", ws.ws_xpixel);
    printf("Pixel height: %d\n", ws.ws_ypixel);

    close(fd);
    return 0;
}
```

4

SBI 是 supervisor binary interface，考虑 ABI，规定了二进制程序间彼此通信的二进制接口，而 SBI 和 ABI 则特权级上有区别，它为操作系统服务，提供了访问底层硬件功能的方式
选择的是 rcore, 用了 `sbi_call`，对应的是 ecall 系统调用的代码；`set_timer`,`console_putchar`,`shutdown` 分别是用了对应的系统调用去执行不同功能，如像 console 输出字节等