shell

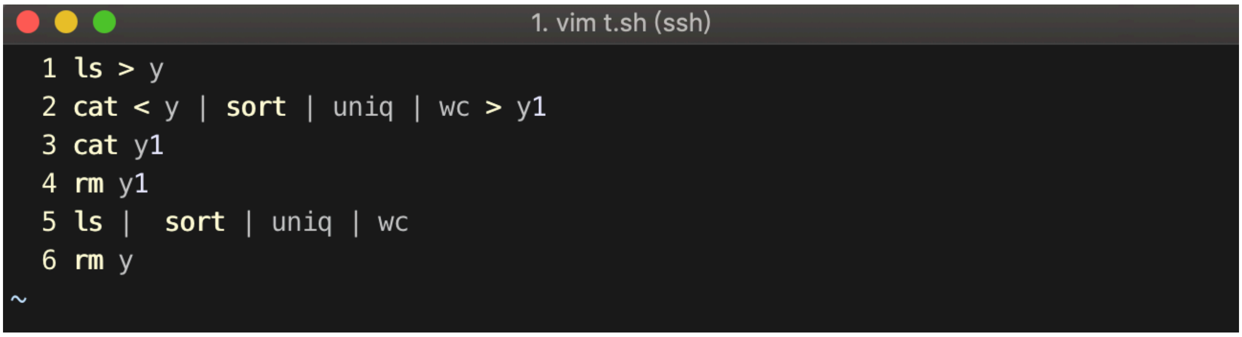
- 姓名：陈姝宇
- 学号：3017218119
- 班级：软工3班

实验内容

实现shell中的可执行命令，重定向命令，以及管道命令

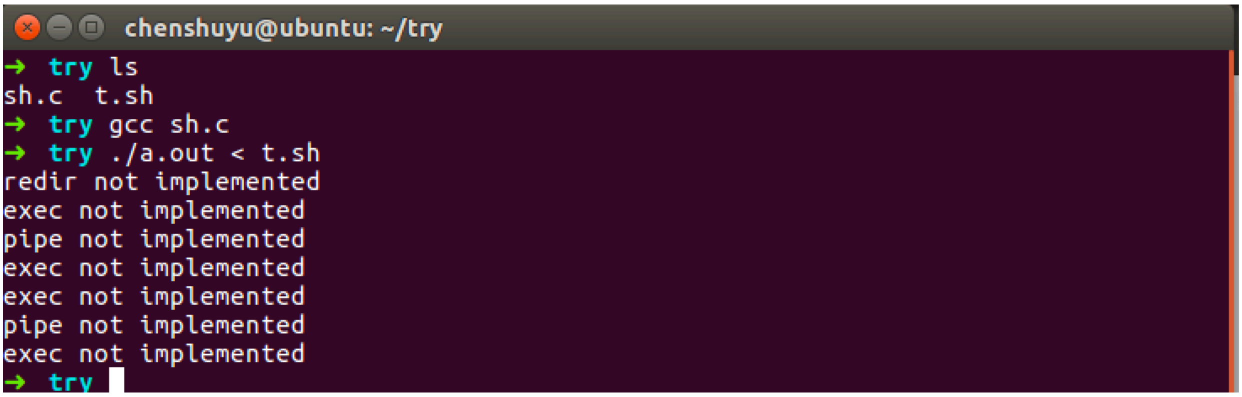
实验步骤

1. 将如下命令加入文件 t.sh 中



```
ls > y
cat < y | sort | uniq | wc > y1
cat y1
rm y1
ls |  sort | uniq | wc
rm y
~
```

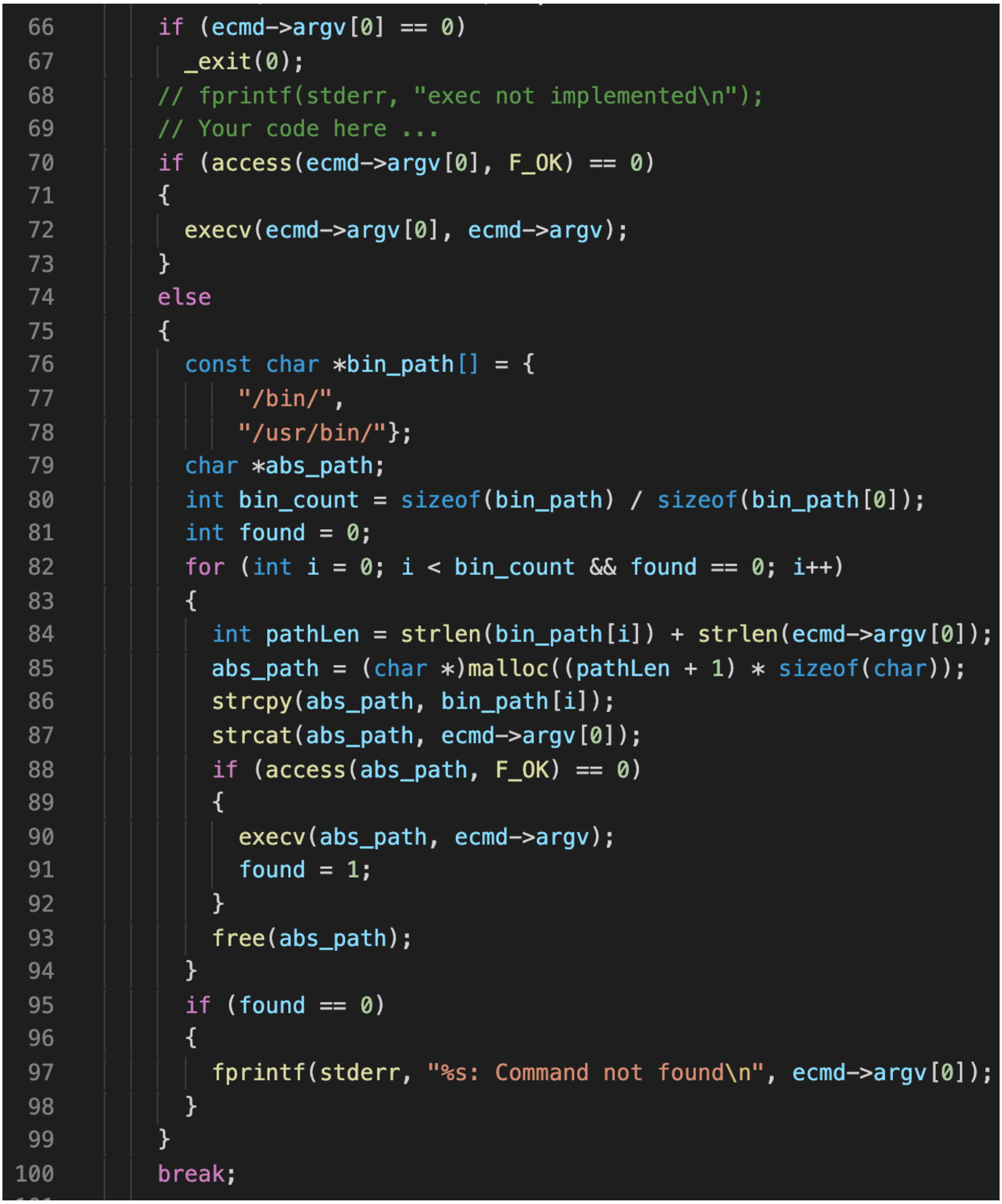
2. 单纯的将文件拷贝进去，并没有完成相应的函数



```
➜  try ls
sh.c  t.sh
➜  try gcc sh.c
➜  try ./a.out < t.sh
redir not implemented
exec not implemented
pipe not implemented
exec not implemented
exec not implemented
pipe not implemented
exec not implemented
➜  try 
```

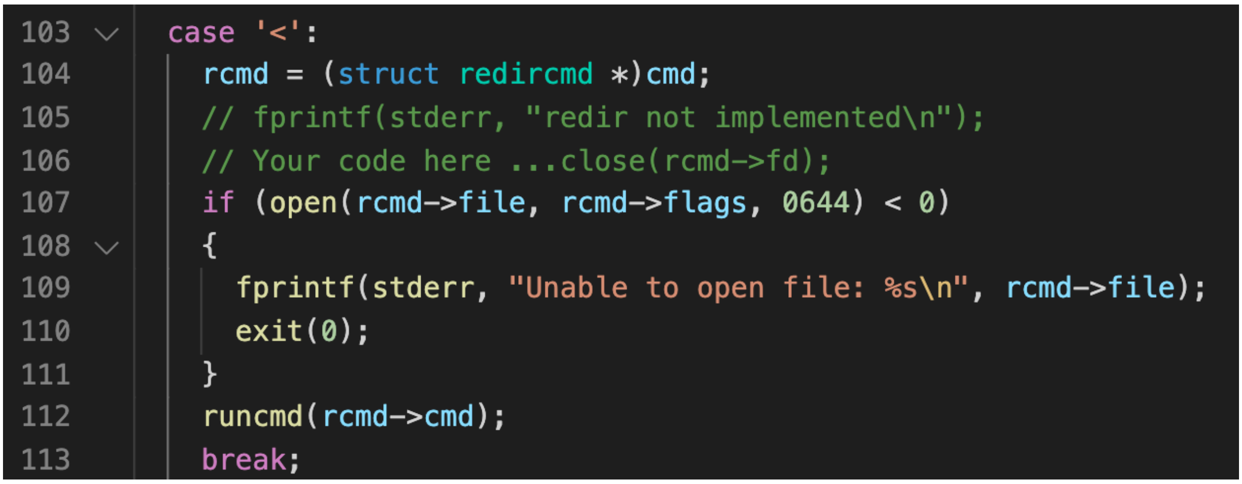
3. 实验各种命令

- 实现可执行



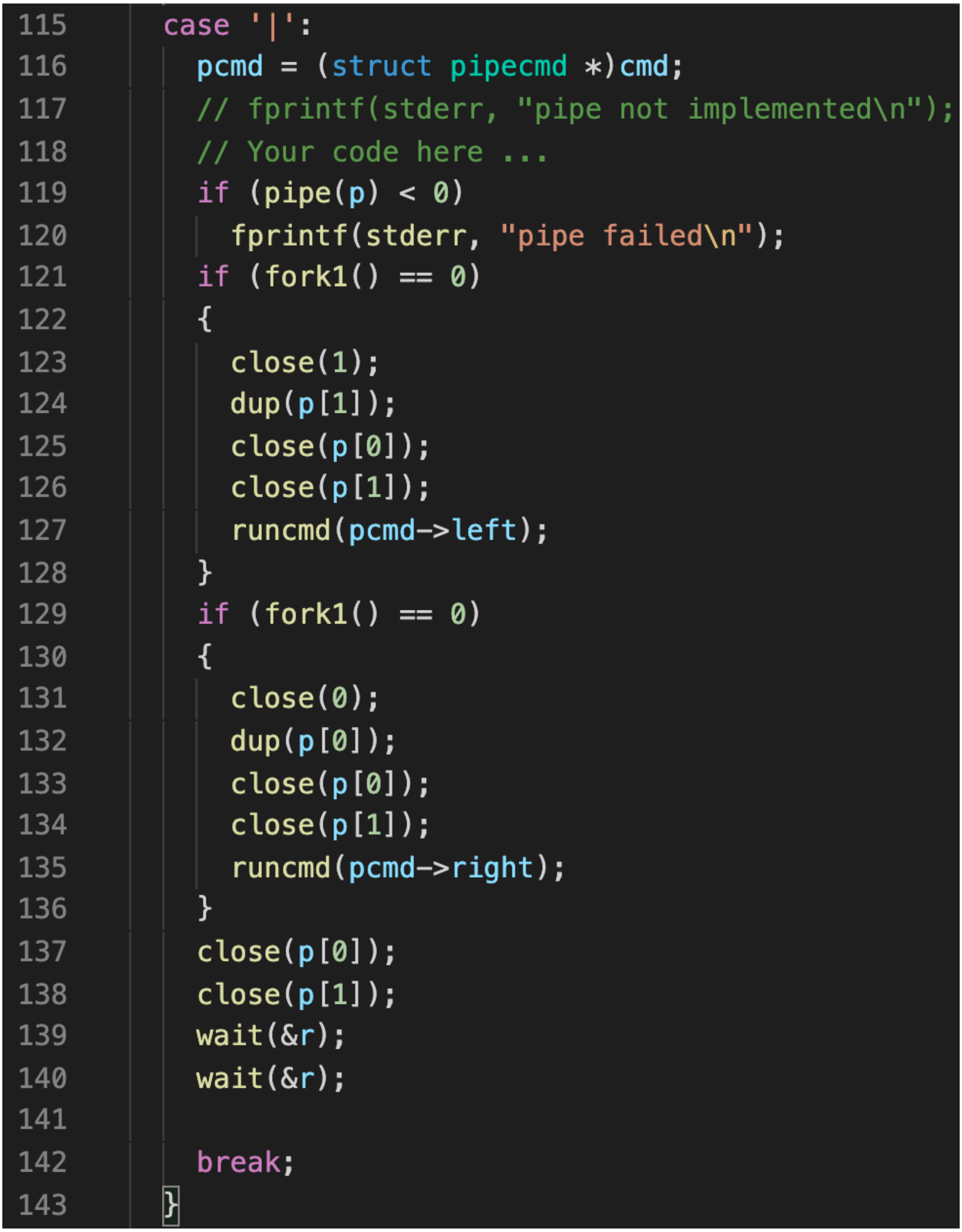
```
    if (ecmd->argv[0] == 0)
      _exit(0);
    // fprintf(stderr, "exec not implemented\n");
    // Your code here ...
    if (access(ecmd->argv[0], F_OK) == 0)
    {
      execv(ecmd->argv[0], ecmd->argv);
    }
    else
    {
      const char *bin_path[] = {
          "/bin/",
          "/usr/bin/"};
      char *abs_path;
      int bin_count = sizeof(bin_path) / sizeof(bin_path[0]);
      int found = 0;
      for (int i = 0; i < bin_count && found == 0; i++)
      {
        int pathLen = strlen(bin_path[i]) + strlen(ecmd->argv[0]);
        abs_path = (char *)malloc((pathLen + 1) * sizeof(char));
        strcpy(abs_path, bin_path[i]);
        strcat(abs_path, ecmd->argv[0]);
        if (access(abs_path, F_OK) == 0)
        {
          execv(abs_path, ecmd->argv);
          found = 1;
        }
        free(abs_path);
      }
      if (found == 0)
      {
        fprintf(stderr, "%s: Command not found\n", ecmd->argv[0]);
      }
    }
    break;
```

- 实现重定向



```
case '<':
  rcmd = (struct redircmd *)cmd;
  // fprintf(stderr, "redir not implemented\n");
  // Your code here ...close(rcmd->fd);
  if (open(rcmd->file, rcmd->flags, 0644) < 0)
  {
    fprintf(stderr, "Unable to open file: %s\n", rcmd->file);
    exit(0);
  }
  runcmd(rcmd->cmd);
  break;
```

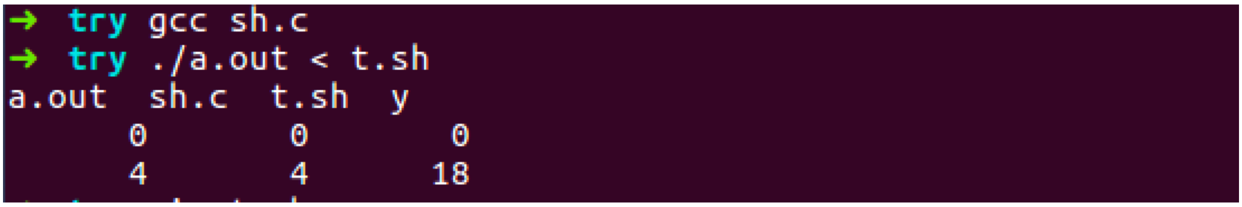
- 实现管道



```
  case '|':
    pcmd = (struct pipecmd *)cmd;
    // fprintf(stderr, "pipe not implemented\n");
    // Your code here ...
    if (pipe(p) < 0)
      fprintf(stderr, "pipe failed\n");
    if (fork1() == 0)
    {
      close(1);
      dup(p[1]);
      close(p[0]);
      close(p[1]);
      runcmd(pcmd->left);
    }
    if (fork1() == 0)
    {
      close(0);
      dup(p[0]);
      close(p[0]);
      close(p[1]);
      runcmd(pcmd->right);
    }
    close(p[0]);
    close(p[1]);
    wait(&r);
    wait(&r);

    break;
  }
```

实验结果



```
➜  try gcc sh.c
➜  try ./a.out < t.sh
a.out  sh.c  t.sh  y
      0       0       0
      4       4      18
```