**实验12 进程控制**

**实验目的**

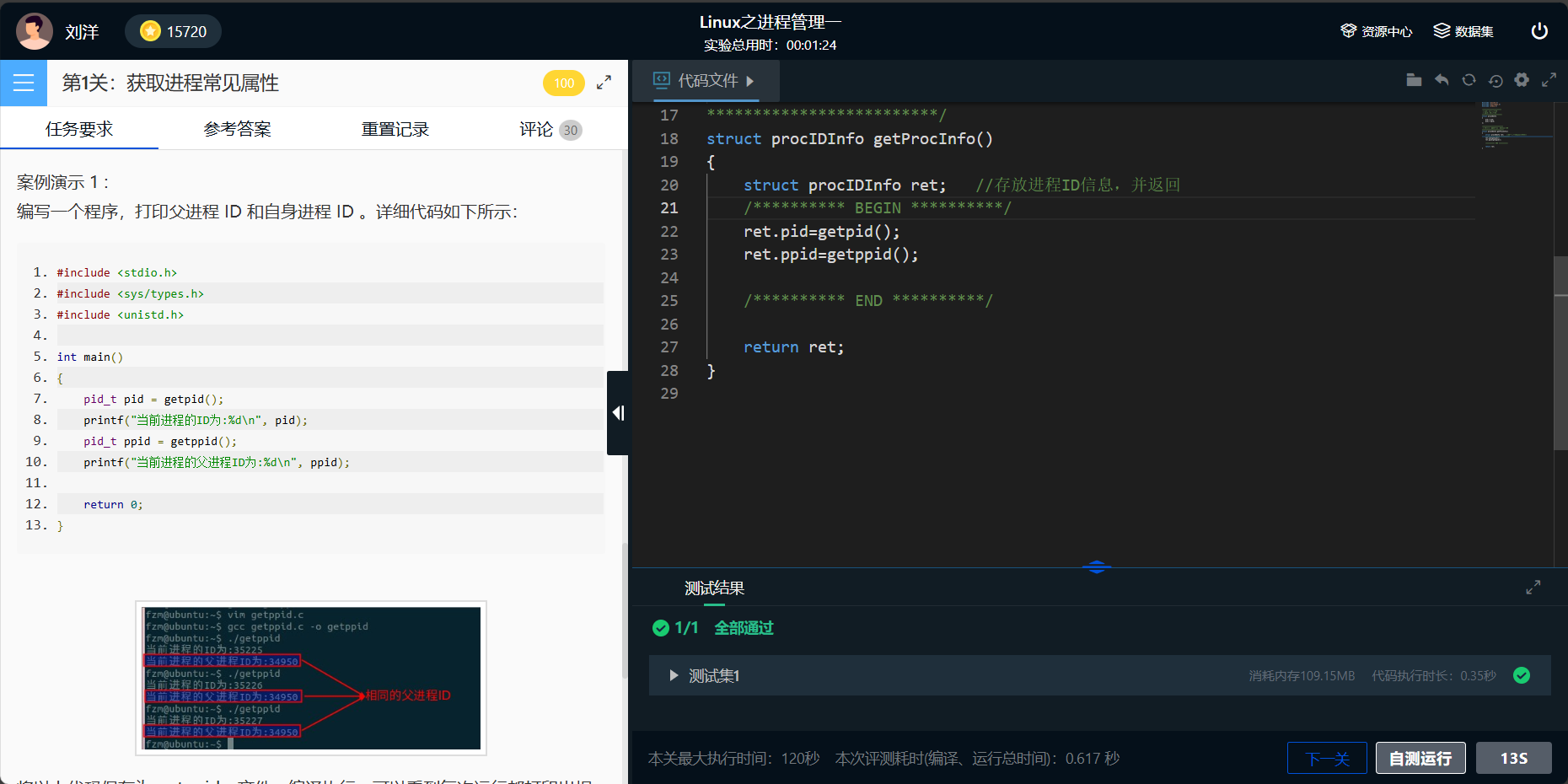
* 1. 掌握常用进程控制函数的用法
  2. 掌握wait和waitpid函数的用法和作用
  3. 掌握处理僵尸进程的方法
  4. 掌握守护进程的特点和程序设计方法

**实验内容**

### 实验7-1进程创建（完成头歌实验，补充命令截图）

**第1关：获取进程常见属性**

1.补全 getProcInfo 函数，用于获取当前进程 ID 和其父进程 ID (提示：将结果存放在procIDInfo结构体中)。

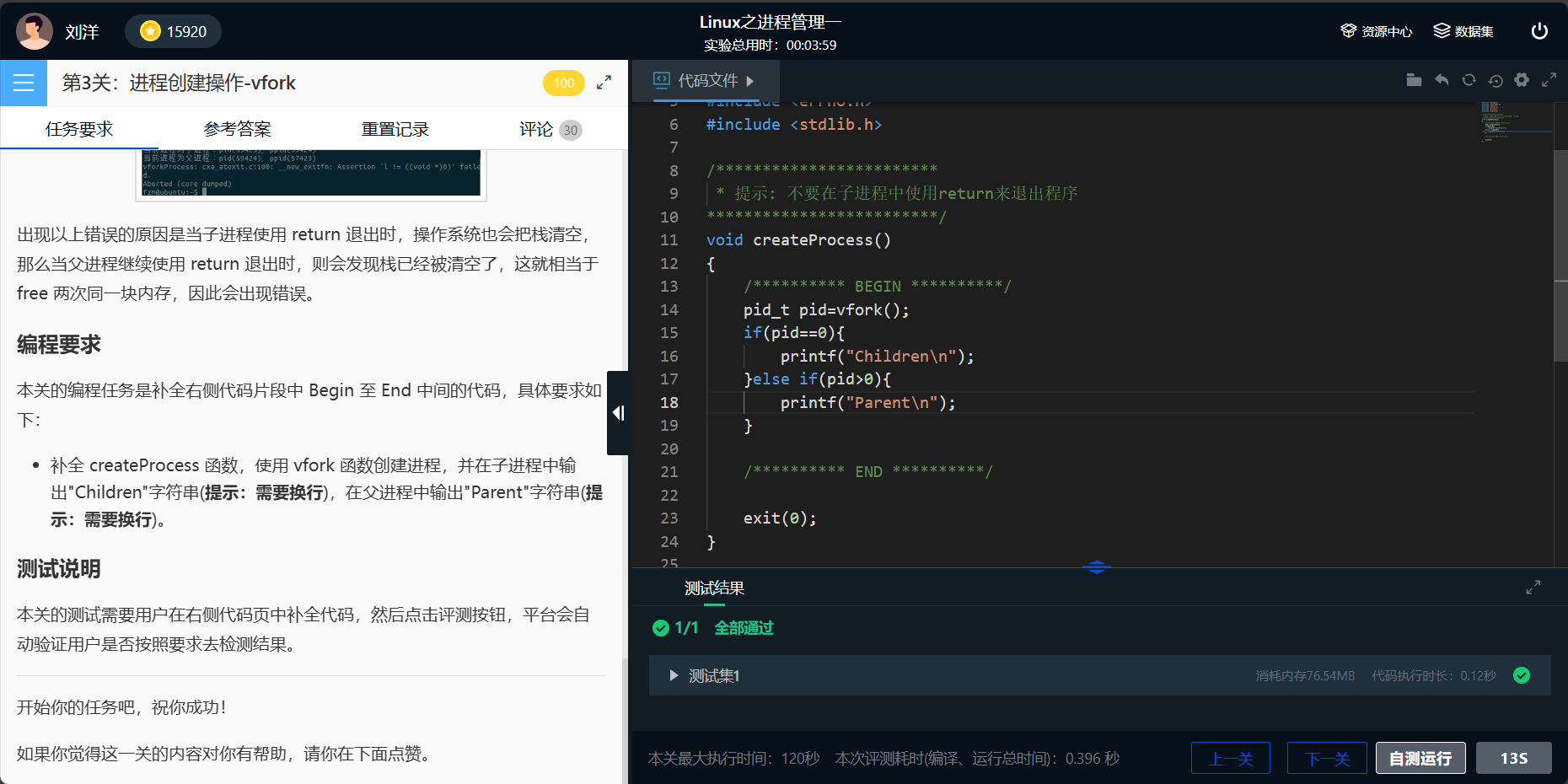


**第2关：进程创建操作-fork**

1.补全 createProcess 函数，使用 fork 函数创建进程，并在子进程中输出 "Children" 字符串，在父进程中输出 "Parent" 字符串。(注意：不要在 createProcess 函数中使用 exit 函数或者 return 来退出程序)。

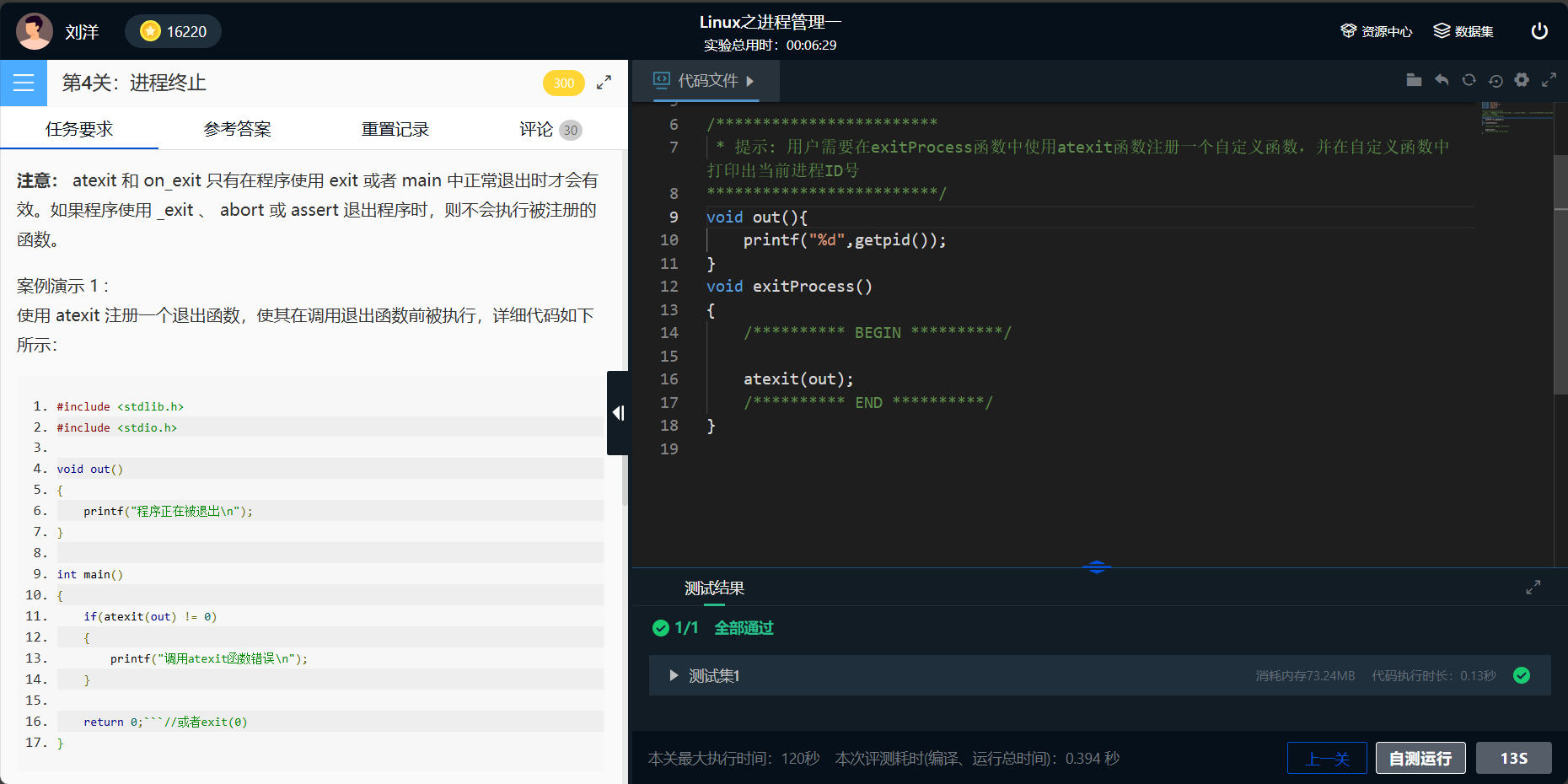
  
**第3关：进程创建操作-vfork**

1. 补全 createProcess 函数，使用 vfork 函数创建进程，并在子进程中输出"Children"字符串，在父进程中输出"Parent"字符串。



**第4关：进程终止**

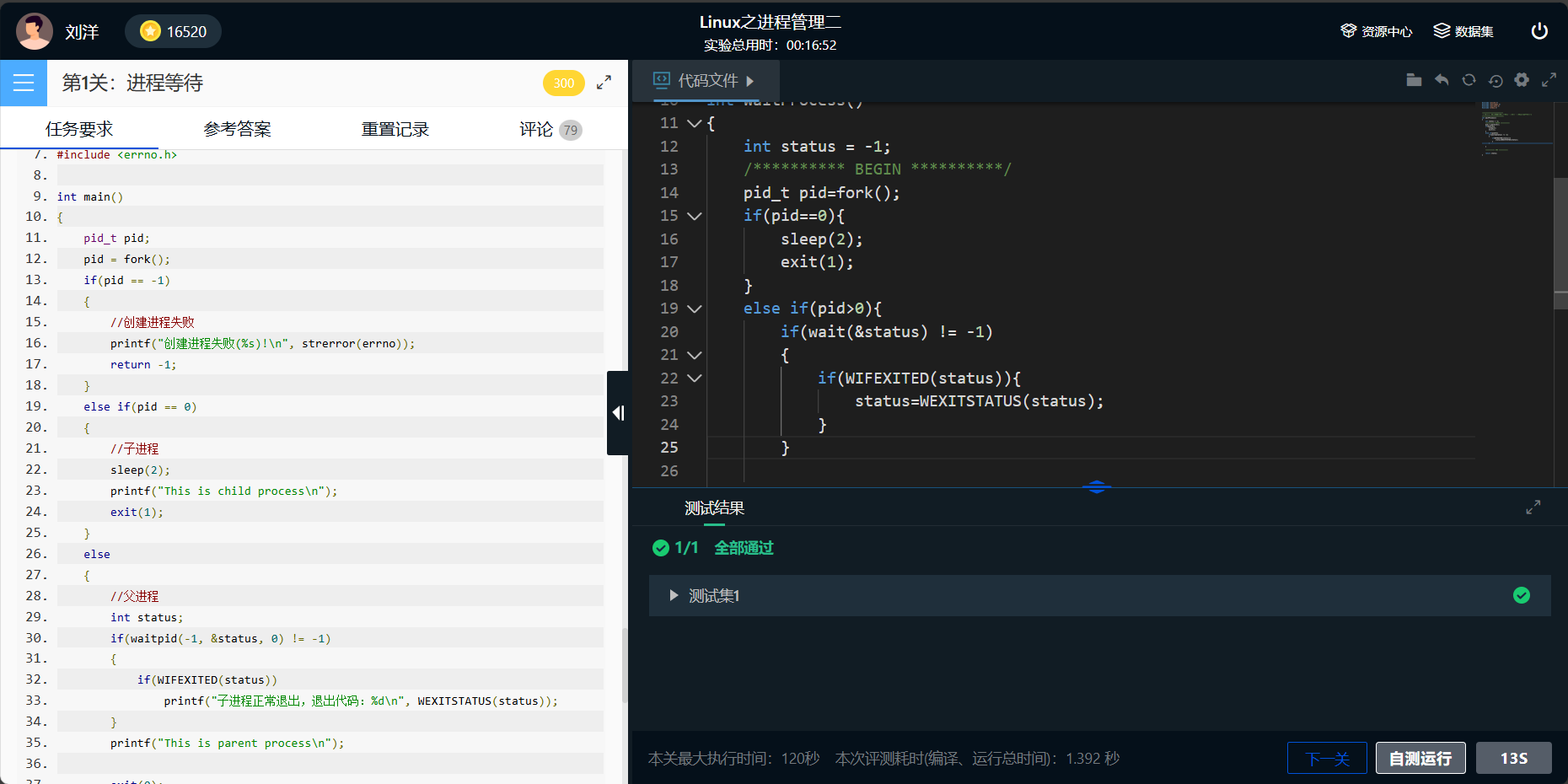
1. 补全 exitProcess 函数，使用 atexit 函数注册一个函数，在注册函数中打印出当前进程的 ID 号。



### 7-2进程控制（完成头歌实验，补充命令截图）

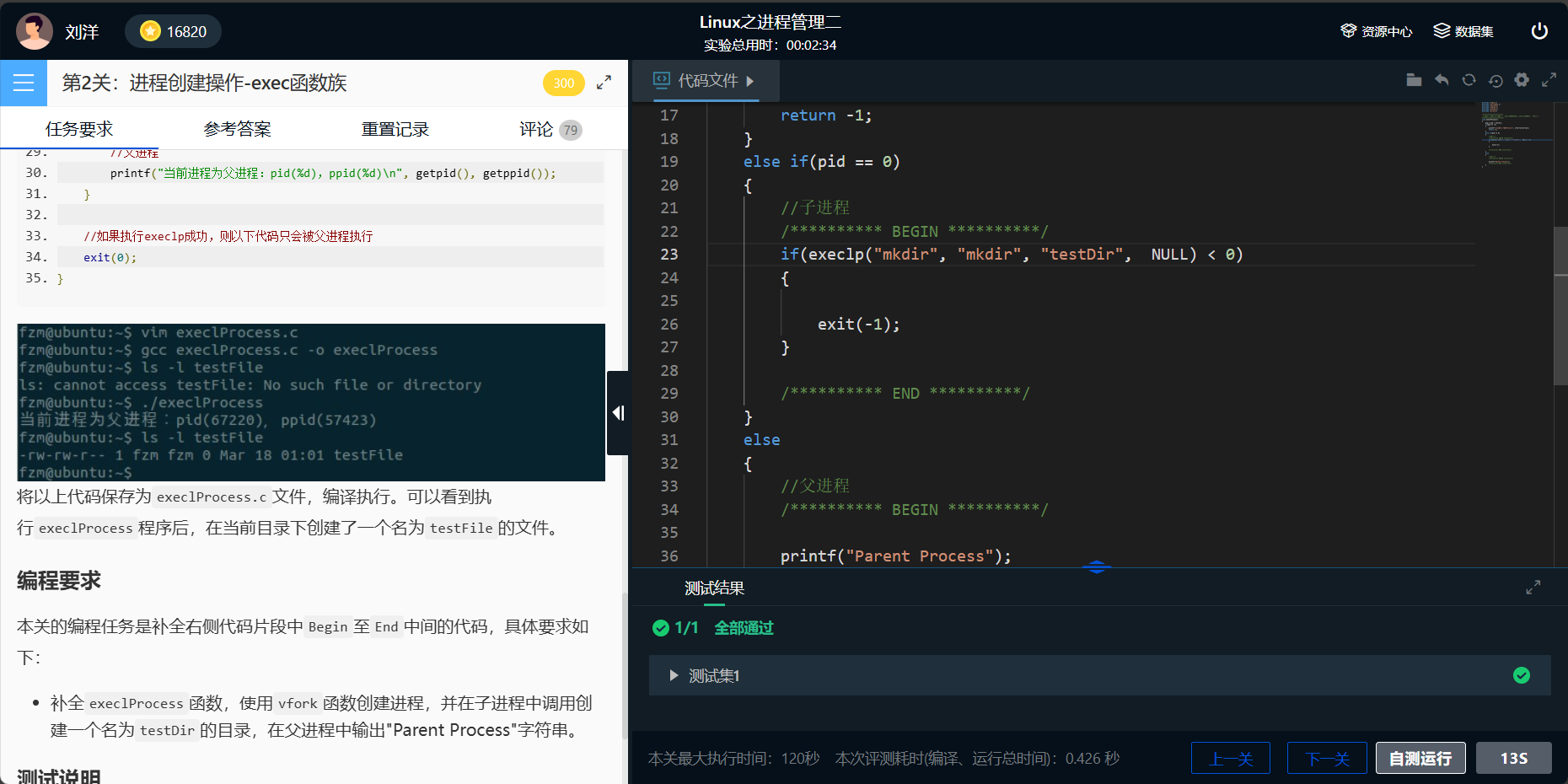
**第1关：进程等待**

补全waitProcess函数，等待子进程结束，并且返回子进程的退出的代码



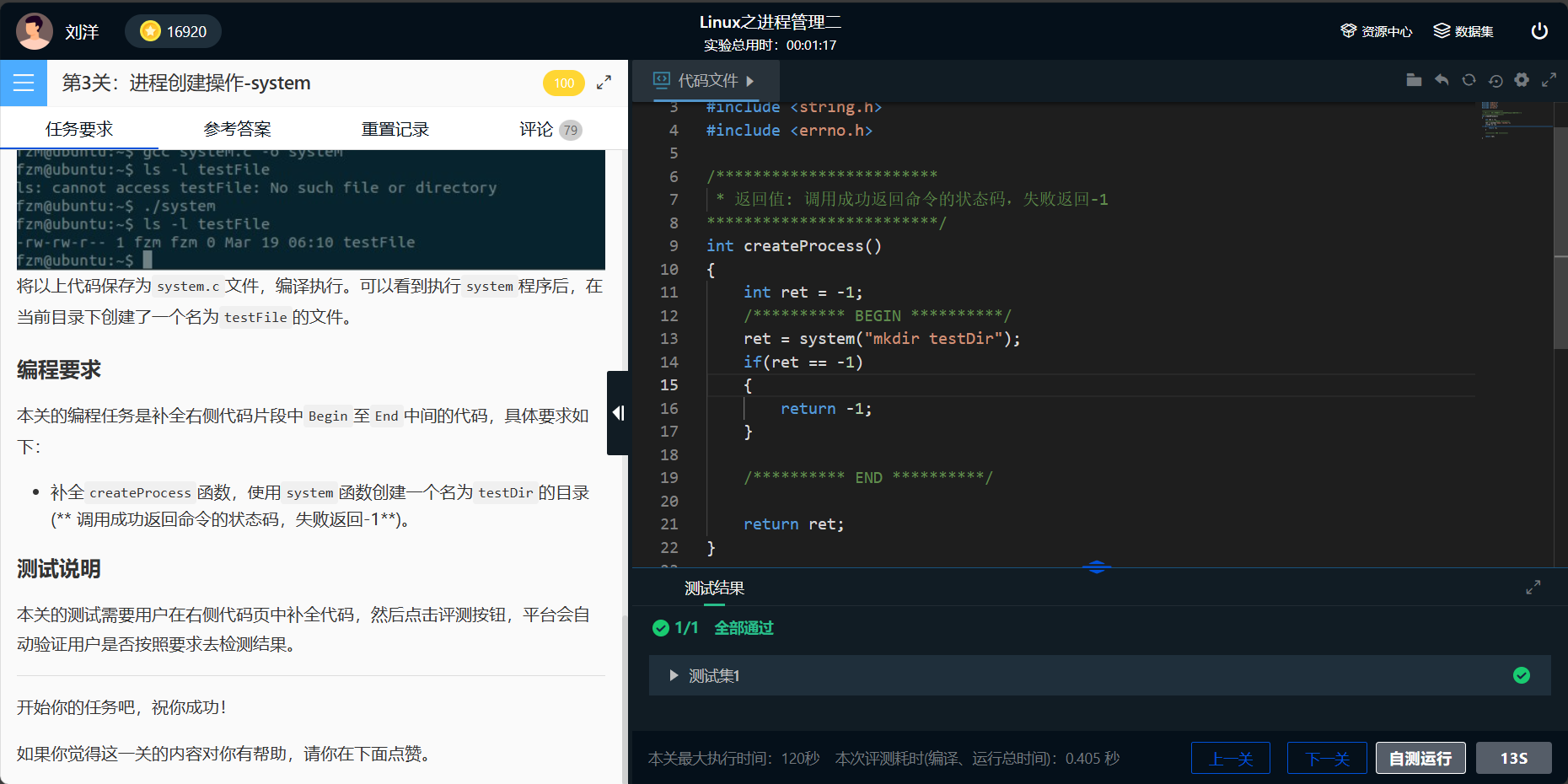
**第2关：进程创建操作-exec函数族**

补全execlProcess函数，使用vfork函数创建进程，并在子进程中调用创建一个名为testDir的目录，在父进程中输出"Parent Process"字符串。



**第3关：进程创建操作-system**

补全createProcess函数，使用system函数创建一个名为testDir的目录(\*\* 调用成功返回命令的状态码，失败返回-1\*\*)。



**三、程序分析与设计**

1.分析以下程序，会有僵尸进程出现吗？后台运行该程序，前台用ps命令查看进程，判断是否出现僵尸进程，如果出现，请用三种方法(wait/waitpid，两次fork，捕捉信号SIGCHLD）修改程序处理僵尸进程。

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

main()

{int pid;

if ((pid=fork())==0)

{ sleep(2);printf("child process %d exit\n",getpid());exit(0);}

else

{ printf("hello child\n");

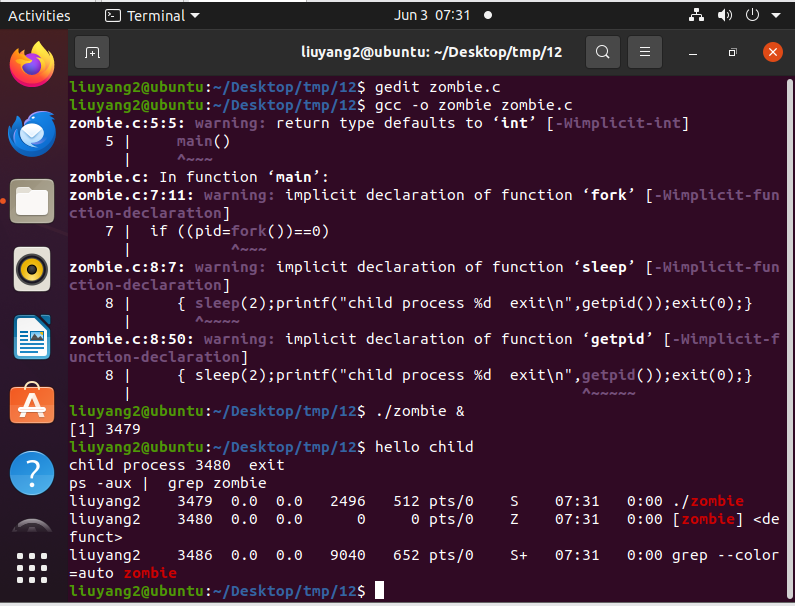
while(1){ sleep(1); };

}

}

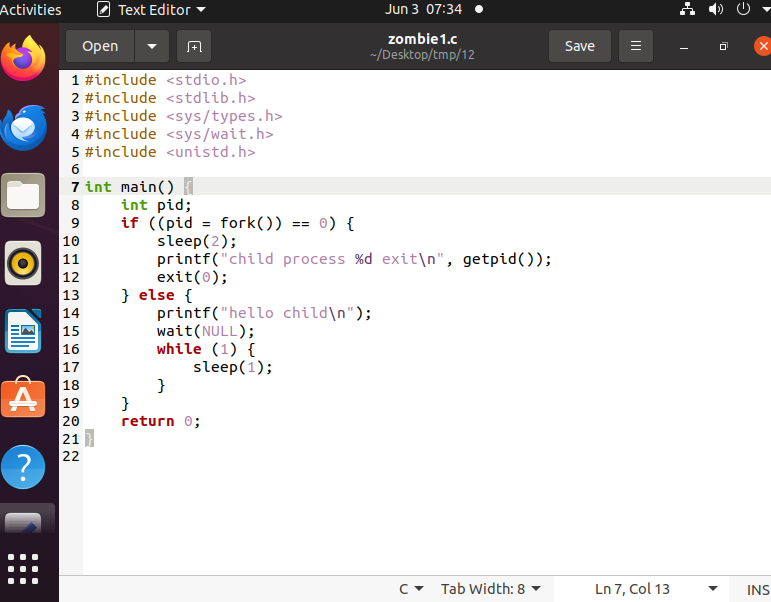
要求:

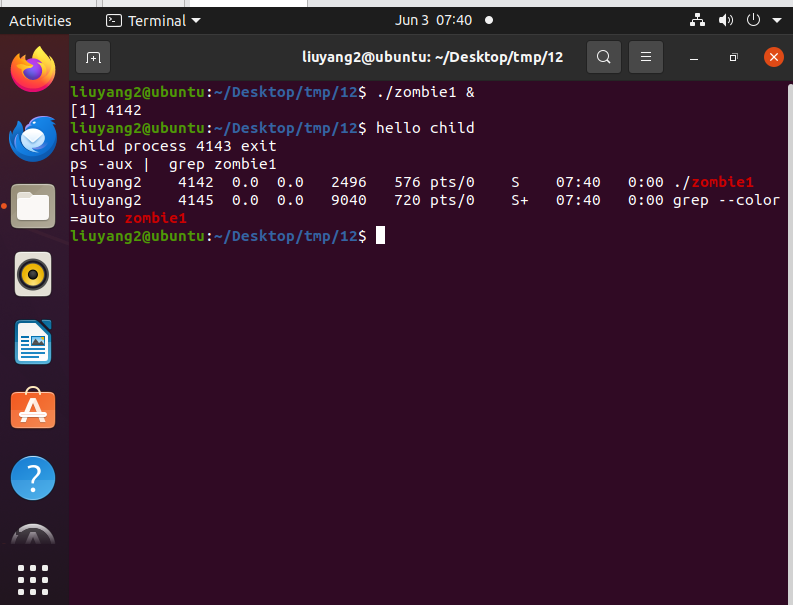
1. 查看僵尸进程的截图



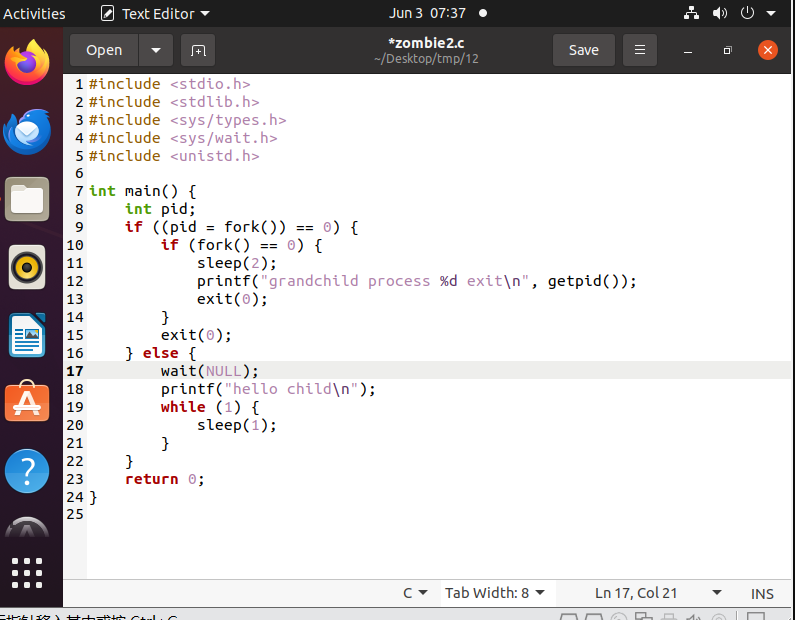
1. 用三种方法处理僵尸进程的程序源代码和运行截图。

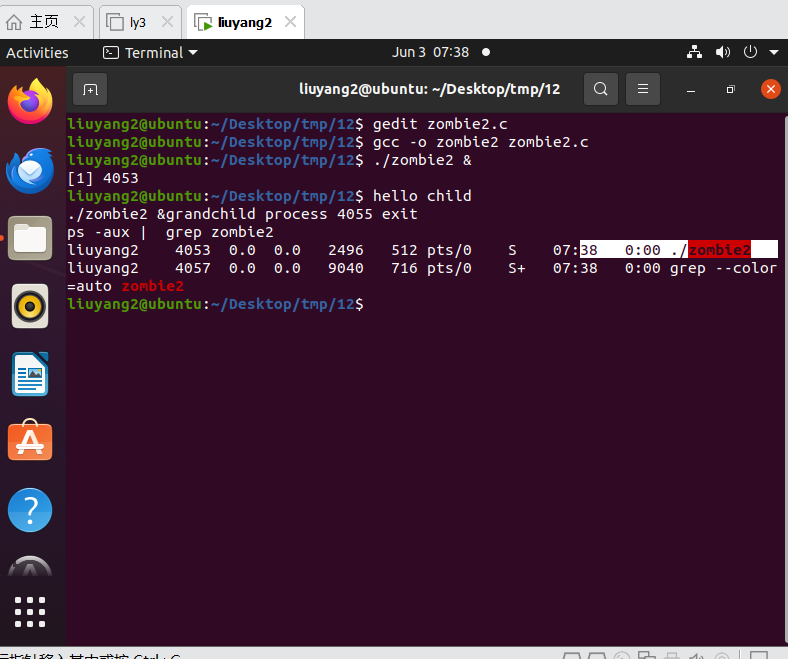
方法一：wait



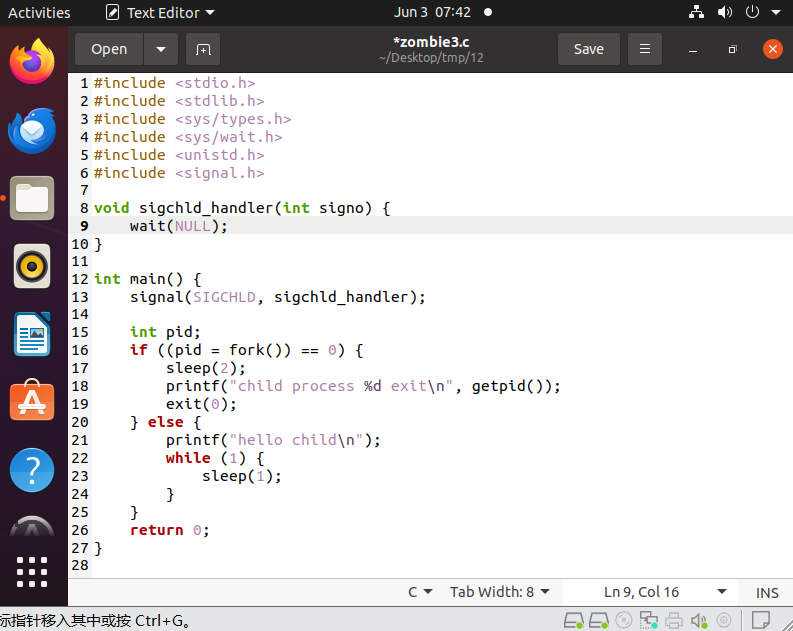


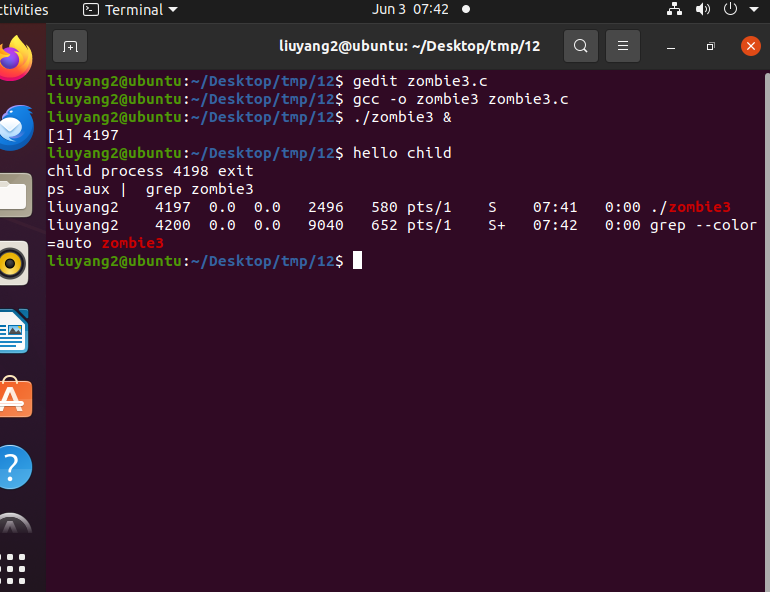
方法二：两次fork





方法三：捕捉信号SIGCHLD





2.编写一程序，要求运行后成为守护进程，复制守护进程的子进程，子进程往某个文件(如test.txt)里写入字符串“测试守护进程：当前系统时间”。父进程等待子进程终止后向系统日志文件“/var/log/messages”写入“子进程终止：当前系统时间”结束，守护进程的错误信息输出到系统日志文件。请把产生守护进程的部分分割成独立的程序文件。提示使用openlog函数和syslog，closelog函数。

要求:

1. 守护进程两个源程序文件代码

查看子进程文件test.txt和系统日志文件/var/log/messages(可用命令tail –f /var/log/messages查看/var/log/syslog)的截图。注意不同版本Linux系统日志文件名不同。

