CSE 2431 至 25

实验室 1

截止日期：2025 年 10 月 2 日星期四晚上 11:30，Carmen

注意：提交作业有半小时的“宽限期”，只要您在午夜之前（晚上 11:59 之前）提交，实验室就会被视为准时；但如果您在午夜或之后提交，无论出于何种原因，实验室都将被视为迟到一天（但是，如果您生病或遇到其他超出您控制范围的情况，请尽早联系讲师，了解延期的可能性）。

1. 目标

本实验将帮助您理解进程的概念、如何创建和终止进程。此外，您还需要熟悉与进程相关的系统调用。

2. 引言

本实验作业要求你构建一个简单的 SHELL 接口（使用 C 编程语言；必须使用 C 语言），该接口接受用户命令、创建（fork）子进程并在子进程中执行用户命令。SHELL 接口为 Shell 用户提供一个提示符，用户可在提示符后输入想要执行的命令。以下示例演示了提示符 CSE2431Sh$ 和用户的下一个命令：cat prog.c。此命令使用 UNIX/LINUX cat 命令在终端上显示文件 prog.c 的内容（假设用户当前工作目录中有一个文件 prog.c）。

CSE2431Shell$ cat prog.c

实现 Shell 接口的一种技术是让父进程首先读取用户在命令行中输入的内容（例如上例中的 cat prog.c），然后创建一个单独的子进程来执行该命令。除非另有说明，否则父进程会等待子进程退出后再继续执行。这在功能上与图 1 中所示的类似。然而，UNIX/LINUX Shell 通常也允许子进程在后台运行（或并发运行）（也就是说，在本例中，父进程在继续执行之前不会等待子进程退出），只需在命令末尾添加与号 (&)。将上述命令重写为

CSE2431Shell$ cat prog.c &

父进程和子进程现在同时运行；也就是说，父进程在分叉子进程后继续执行，并且不等待子进程退出（这意味着父进程和子进程同时运行，因为父进程没有等待子进程；请注意，即使父进程和子进程同时运行，调度程序实际上也会准确确定何时运行每个进程）。

使用 fork() 系统调用创建单独的子进程，并使用以下方式执行用户的命令
exec() 系列中的系统调用之一（对于本实验，子程序将调用/调用 execvp()；有关此系统调用的更多详细信息，您可以使用 man 命令获取在线文档。您应该使用 man 查看命令文档以确定如何将参数传递给 execvp()）。

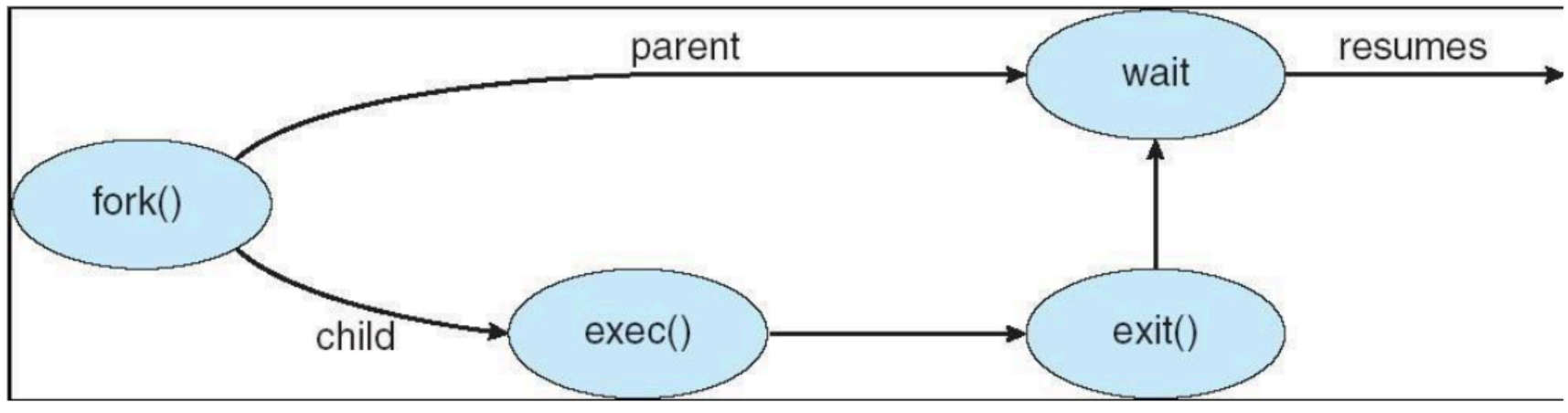

图 1. 进程创建和终止

3. 一个简单的 Shell

首先，为了完成本实验，你需要创建一个 cse2431 目录/文件夹，然后在 cse2431 目录下创建一个 lab1 目录/文件夹，用于完成 coelinux 上的作业。shellA.c 文件提供了一个提供命令行 shell 基本代码的程序，你可以在 Carmen 的“文件”>“实验室”>“实验室 1”文件夹中找到它。在 Carmen 上获取用于完成作业的 shellA.c 文件及其代码框架。要从 Carmen 获取该文件，请登录 coelinux，然后在 shell 提示符下启动 Firefox 浏览器，如下所示：

$ firefox https://carmen.osu.edu/#

浏览器启动后，您可以登录 Carmen，然后依次转到 Files、课程的 Lab 文件夹和 Lab 1 文件夹，并将 shellA.c 文件下载到您的 coelinux ~/Downloads 目录（默认情况下，它会在 coelinux 中下载到此目录）。您可以使用 cp 命令（先 cd 到 cse2431，再 cd 到 lab1）将 shellA.c 文件从 ~/Downloads 复制到您的 cse2431/lab1 目录。即，运行：

$ cd ~/cse2431

$ cp ~/Downloads/shellA.c ./

shellA.c 程序由两个函数组成：main() 和 setup()。setup() 函数读取用户的下一个命令（最多 40 个字符），然后将其解析为单独的标记，用于填充待执行命令的参数向量。（如果命令要在后台运行，则以“&”结尾，setup() 将更新参数 background，以便 main() 函数能够相应地执行操作。当用户输入以下内容时，shellA.c 程序终止。在命令行提示符下，setup() 然后调用 exit()（一个将终止运行 shell 的进程的系统调用）。

main() 函数会显示提示符 CSE2431Sh$，然后调用 setup()，等待用户输入命令。用户输入的命令内容会加载到 args 数组中。例如，如果用户在 CSE2431Sh$ 提示符下输入 ls -l，则 args[0] 将指向字符串 ls，而 args[1] 将指向字符串 -l。（“字符串”是指以空字符结尾的 C 语言字符串变量。）

```
＃包括
＃包括

#定义最大线 40

/** setup() 读取存储在 inputBuffer 中的下一个命令行字符串，并使用空格作为分隔符将其分成不同的标记。setup() 修改 args 参数，使其包含指向空终止字符串的指针（这些字符串是最新用户命令行中的标记）以及一个 NULL 指针，指示参数列表的结尾，该指针位于已分配给 args 的字符串指针之后。*/



void setup（char inputBuffer []，char *args []，int
*background）{
/** 完整代码可在文件 shellA.c 中找到 */} int main(void) {char
inputBuffer[MAX_LINE];/* 用于保存输入命令的缓冲区 */int
background;/* 如果命令后跟 '&'，则等于 1 */char
*args[MAX_LINE/2+1];/* 命令行参数 */pid_t ret_val;/* 用于保存 fork
返回的值 */while (1){



        背景=0；
        printf("CSE2431Shell$ ");
       setup(inputBuffer,args,&background); /* 获取下一个命令 */
       /* 步骤如下：
         （1）使用 fork() 创建一个子进程
         （2）如果 fork() 出错，则打印错误消息并退出
         （3）子进程将使用适当的参数调用 execvp()
(4）如果 background == 0，父级将等待，否则返回 while 循环顶部以调用 setup() 函数。 */ } }
```

本实验作业要求你创建一个子进程并执行用户输入的命令。为此，你需要修改 shellA.c 中的 main() 函数，使其在 setup() 返回后分叉一个子进程。之后，该子进程将执行 shell 用户指定的命令。

如上所述，setup() 函数会将 args 数组的内容与 char * 类型的指针一起加载到用户输入的命令字符串中（也就是说，setup 将命令分成一系列由空格分隔的字符串，并在 args 数组中放置一个 char * 类型的指针，该指针指向命令中每个由空格分隔的字符串）。您无需编写任何代码来解析用户命令，也不应该尝试这样做！setup() 会为您完成这项工作；请勿修改 setup() 函数中的代码！args 数组中的参数将传递给 execvp() 函数，该函数具有以下接口：

execvp（char *命令，char *参数[]）；

其中 command 代表要执行的命令（该命令实际上是磁盘上文件中的库程序（的名称），它可以作为进程运行以执行用户在命令行提示符下输入的命令；所有 Linux 命令（cd、cp、ls、ps、cat 等）都是程序），params 存储用户在命令行上输入的该命令的参数。您可以通过发出命令“man execvp”找到有关 execvp() 的更多信息。请注意，您应该检查在 main() 中编写的代码中的 background 值，以确定父进程是否等待子进程退出。要让父进程等待，您可以使用 wait() 系统调用。

4. 说明

你需要在我们的 CSE Linux 环境 coelinux 中完成你的解决方案。请使用 gcc 编译器（如果你想了解编译器，请运行“man gcc”）。评分员将在此环境下使用以下编译器编译并测试你的解决方案：

```
$ gcc shellA.c -o shellA
```

任何无法编译的程序都将得到零分，此规则没有例外；在提交代码之前，请务必确认其能够编译并正确运行。评分员不会花时间修复你的代码（即使是拼写错误等最小错误），因为你在最后一刻（甚至不是最后一刻）引入了一些简单的错误。你有责任留出足够的时间，确保你的代码能够在截止日期之前编译、运行和测试完毕，并在截止日期当天晚上 11:30 之前提交（如上所述，只要你在 11:59:59 之前提交，你的实验就不会被视为迟交，但不要等到最后一刻；请提前开始工作、完成并提交实验，如果可能的话至少提前一天，或者最迟在实验截止日期当天下午 6:00 或 7:00 之前提交，以避免出现问题）。帮自己一个忙，不要给自己留太少的时间来提前完成作业，以免在最后一刻做出更改，从而面临在提交前没有时间编译和测试的风险；如果你这样做，不幸的是，任何后果都是你自己的责任（这是专业环境中的工作方式，所以我们希望鼓励每个人都认识到这一点，并按照专业人士的期望来工作）。

父子进程的同步：在标准的 Unix/Linux shell 中，如果子进程并发运行，父进程会返回到 while 循环的顶部，并经常在子进程写入输出之前（或者在子进程输出的中间；例如，如果父进程没有等待子进程，您可能会在子进程输出之前看到 shell 提示符）。这是 Unix/Linux shell 中的正常行为，如果您的 shell 代码有这种行为，您不必担心，因为这是正常的。

5. 测试用例和警告

确保您的 shell 在以下测试用例中正确执行（从命令行在 Linux shell 中运行这些命令以查看它们的作用）：

```
CSE2431Shell$mkdir new

CSE2431Shell$ ls

CSE2431Shell$ ls -al

CSE2431Shell$ 日期

CSE2431Shell$ 猫壳 A.c
```

```
CSE2431Shell$ ls &

CSE2431Shell$ ls -al &

CSE2431Shell$ 日期 &
```

警告：请确保不要尝试执行更改目录的命令（例如
cd、cd .. 等等）！另外，请确保不要尝试使用输入重定向来测试你的 shell 程序（我们不会用这种方式测试）！也就是说，你必须输入上面显示的测试命令来测试你的 shell。你不能把它们放在输入文件中，然后使用输入重定向。

我们将根据您的解决方案以及代码在所提供测试中的表现进行评分。您编写的任何代码（不包括您收到的代码）都应以专业的方式进行注释。

6. 提交

您应该将您的 shellA.c 文件提交给 Carmen Canvas（警告：请确保您不要提交
可执行文件 - 仅提交 AC 源文件 - 如果您
提交可执行文件！提交前请勿压缩文件。您可以在 coelinux 命令行提示符下启动 Firefox 浏览器提交（如果您的提示符不是 $，则没有影响）：

```
$ firefox https://carmen.osu.edu/#
```

Carmen 打开后，您可以登录并前往我们课程的 Lab1 作业，提交您的 shellA.c 源文件。只需提交您的 shellA.c 源文件即可。

不要提交文件夹！只提交 shellA.c 源文件！

注意：在文件“shellA.c”的开头，你需要添加一条注释，告诉评分者你的完整
名称。在适当的地方添加其他注释，以记录您添加到 shellA.c 源代码文件的任何代码，但您不需要为您未编写或修改的代码添加注释。

对于在本文件顶部给出的截止日期之后、但在上述截止日期和时间之后 24 小时内提交的任何实验，将被处以 25% 的滞纳金；对于在上述截止日期和时间之后 24 小时内提交的任何实验，将被处以 25% 的滞纳金。
逾期提交不计学分。实验作业必须由学生独立完成。注意：避免多次提交（提交次数没有上限，但多次提交不专业；提交前务必仔细并彻底测试，否则通常无需重新提交）

8. 广场

Piazza 旨在为学生提供一个讨论平台。建议的实验主题包括：语法和/或执行错误、逻辑错误、系统调用相关问题等。通常情况下，一个学生的问题实际上是其他几个学生的问题，所以这篇文章对每个人都有帮助。欢迎随时
互相回答问题并讨论答案等。讲师还将监控发布的答案
在 Piazza 上发布帖子以确保其正确性，但其目的是让学生在课堂之外有一个可以讨论与课程相关主题的平台。Piazza 的帖子和问答也可以为后续课程中可能遇到类似问题的学生提供很好的参考。请勿在问题中发布代码或关于如何实现解决方案的想法，除非您将问题设置为私密（默认情况下将设置为私密）。但是，如果您不发布代码或关于解决方案的想法，请更改您的

请在帖子中设置非私密状态，以便其他学生能够看到已发布的问题和答案。如果您不这样做，讲师会在发布答案之前将您的帖子更改为对其他人可见。