练习 01

对于本练习,您将比较系统调用和您可能更习惯使用的库例程。我已经为您编写了一个简短的程序,writeLibc.c。您可以在 Moodle 的课程主页上找到它。该程序使用 C 标准库创建输出文件 output.txt。它使用 fputc() 向文件写入超过 1,000,000 个字符,一次一个字符,定期倒回到文件的开头,因此它不会变得太大。

编译这个程序并在其中一台 EOS linux 机器上计算它的执行时间。以下命令应该完成这项工作:

$ gcc -Wall -std=c99 writeLibc.c -o writeLibc $ time ./writeLibc

完成此程序后,删除它创建的输出文件。这将帮助您确保在下一部分不会犯错:

$ rm -f 输出.txt

让我们将此程序与直接使用系统调用接口写入输出文件的程序进行比较。将 writeLibc.c 复制到名为 writeSystem.c 的文件中。我将总结将程序转换为使用系统调用写入文件需要执行的操作,但您还需要自己阅读一些文档。您可以使用 man 命令阅读 open() 等系统调用的联机文档。下面的 2 选项表示从在线手册的第 2 部分获取文档,其中记录了系统调用:

$ 男人 2 打开

- 因为您将使用系统调用接口,所以您将使用文件描述符(一个整数)来跟上文件,而不是文件指针。 · 当您开始在程序中放置系统调用时,您将需要在顶部包含更多的标题。每个系统调用的联机文档都会告诉您需要包含哪些标头。

- 将对fopen() 的调用替换为对open() 系统调用的调用。它期望文件名作为第一个参数,各种选项的按位或作为第二个参数。您将需要按位或组合选项,这些选项表示您想要打开文件进行写入,并且如果文件不存在则您想要创建该文件。在创建文件时,您需要提供一个可选的第三个参数。它允许您为新文件指定权限。像 0600 这样的八进制值是一个不错的选择;它赋予所有者(您)读取和写入文件的权利。 · 将对fputc() 的调用替换为对write() 系统调用的调用。 Write 会让你写出尽可能多的

  给定一个字节数组(即指针)和它应该写入的字节数,你想要的字节数。对于本练习,您只需一次写出一个字节。您需要将要写入的字节的地址作为第二个参数传递给 write()。

- 将对rewind() 的调用替换为对lseek() 的调用。使用常量 SEEK_SET 作为最后一个参数将使倒回文件的开头变得容易。

- 将对fclose() 的调用替换为对close() 系统调用的调用。

完成后,编译此版本的程序并计算其执行时间:

$ gcc -Wall -std=c99 writeSystem.c -o writeSystem $ time ./writeSystem

您可能会发现此版本的程序运行速度比使用 libc 的版本慢得多。起初,这可能令人惊讶,因为 writeSystem 直接进行系统调用,而 writeLibc 通过库进行。
性能上的差异是因为进行系统调用比仅调用子例程要昂贵得多。该程序的 writeSystem 版本确实必须进行超过一百万个系统调用,每个系统调用都有一点运行时开销。在 writeLibc 中,我们对库函数 fputc() 进行了一百万次调用。

libc 库最终仍然需要进行系统调用才能将字符实际写入文件,但 libc 足够聪明,可以在进行系统调用以写出多个字符之前缓冲大量单字符写入
一次。

如果你想看到区别,你可以通过 strace 运行这两个程序。这将报告任一程序进行的每个系统调用。你可能想在它们完成之前用 ctrl-C 杀死它们中的每一个,因为它们都会产生大量的输出。在你杀死它们之前,你应该看到 writeSystem 对每个写入的字符进行系统调用,但是 writeLibc 缓冲字符并且实际上调用 write() 的频率较低(对于这个程序,只有当它倒带时)。

$ strace ./writeLibc <大量输出>
$ strace ./writeSystem <大量输
出>

完成后,上交已完成的 writeSystem.c 的源代码。将文件提交给名为 EX01 的 Gradescope 作业。