系統程式 HW3

410410060 資工二 林柔均

1. 請問 buffersize 分別是：0、-1、4KB、16KB、64KB、1MB、8MB的執行速度分別為何？

參數測試

Buffer_size: 0

```
~/De/h/Sy/hw4 ❯ time ./fileperf input.txt output.txt 0                16:59:15

================
CPU     99%
user    1.518 s
system  12.905 s
total   14.439 s
```

Buffer_size: -1

```
~/De/h/Sy/hw4 ❯ time ./fileperf input.txt output.txt -1       14s 17:01:06

================
CPU     97%
user    0.234 s
system  0.655 s
total   0.913 s
```

Buffer_size: 4KB = 4096 B

```
~/De/h/Sy/hw4 ❯ time ./fileperf input.txt output.txt 4096             17:01:14

================
CPU     90%
user    0.225 s
system  0.030 s
total   0.282 s
```

Buffer_size: 16KB = 16484 B

```
~/De/h/Sy/hw4 ❯ time ./fileperf input.txt output.txt 16384            17:01:31

================
CPU     88%
user    0.175 s
system  0.017 s
total   0.216 s
```

Buffer_size: 64KB = 65536 B

```
 ~/De/h/Sy/hw4 ❯ time ./fileperf input.txt output.txt 65536          17:01:57

================
CPU     77%
user    0.159 s
system  0.030 s
total   0.243 s
```

Buffer_size: 1MB = 1048576 B

```
 ~/De/h/Sy/hw4 ❯ time ./fileperf input.txt output.txt 1048576        17:02:16

================
CPU     88%
user    0.167 s
system  0.027 s
total   0.218 s
```

Buffer_size: 8MB = 8388608 B

```
 ~/De/h/Sy/hw4 ❯ time ./fileperf input.txt output.txt 8388608        17:02:33

================
CPU     80%
user    0.155 s
system  0.024 s
total   0.222 s
```

結論

Buffer_size 越大，程式速度越快。但當數值大到一定程度時加速效果變小，差異不大

2.使用 ltrace 觀察應用程式呼叫

我的 ArchLinux 裡沒有 ltrace 指令，要自行下載: `yay -S ltrace`

Buffer_size: 0 (unbuffered)

```
 ~/De/homework/SystemProgramming/hw4 ❯ ltrace -c ./fileperf input.txt output.txt 0
% time     seconds  usecs/call     calls      function
------ ----------- ----------- --------- --------------------
 82.53   29.642686          51    572463 fgetc
 17.31    6.219218          66     93804 fputs
  0.16    0.056003          66       846 fputc
  0.00    0.000466         233         2 fopen
  0.00    0.000304         152         2 setvbuf
  0.00    0.000205         205         1 atoll
  0.00    0.000180          90         2 fclose
------ ----------- ----------- --------- --------------------
100.00   35.919062                667120 total
```

Buffer_size: -1 (linebuffered)

```
 ~/De/homework/SystemProgramming/hw4 ❯ ltrace -c ./fileperf input.txt output.txt -1
% time     seconds  usecs/call     calls      function
------ ----------- ----------- --------- --------------------
 83.61   19.669146          34    572463 fgetc
 16.03    3.770635          40     93804 fputs
  0.36    0.083953          99       846 fputc
  0.00    0.000291         145         2 fopen
  0.00    0.000102          51         2 fclose
  0.00    0.000080          40         2 setvbuf
  0.00    0.000065          65         1 atoll
------ ----------- ----------- --------- --------------------
100.00   23.524272              667120 total
```

Buffer_size: 4096

```
 ~/De/homework/SystemProgramming/hw4 ❯ ltrace -c ./fileperf input.txt output.txt 4096
% time     seconds  usecs/call     calls      function
------ ----------- ----------- --------- --------------------
 85.68   19.008860          33    572463 fgetc
 14.18    3.146580          33     93804 fputs
  0.13    0.028371          33       846 fputc
  0.00    0.000533         266         2 fopen
  0.00    0.000132          66         2 setvbuf
  0.00    0.000126          63         2 fclose
  0.00    0.000092          92         1 atoll
------ ----------- ----------- --------- --------------------
100.00   22.184694              667120 total
```

3. 使用 strace 觀察你的應用程式呼叫「作業系統的情況」

strace 也要自行下載: `yay -S strace`

Buffer_size: 0 (unbuffered)

```
 ~/Desktop/homework/SystemProgramming/hw4 ❯ strace -c ./fileperf input.txt output.txt 0
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 66.73    1.146158           2    572464           read
 33.26    0.571308           6     94650           write
  0.01    0.000123          30         4           close
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         3           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         2           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0         4           openat
  0.00    0.000000           0         2           newfstatat
  0.00    0.000000           0         1           set_robust_list
  0.00    0.000000           0         1           prlimit64
  0.00    0.000000           0         1           getrandom
  0.00    0.000000           0         1           rseq
------ ----------- ----------- --------- --------- ----------------
100.00    1.717589           2    667150         2 total
```

Buffer_size: -1 (linebuffered)

```
~/De/homework/SystemProgramming/hw4 ❯ strace -c ./fileperf input.txt output.txt -1
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 98.68    0.052162           4     10692           write
  1.01    0.000533           3       142           read
  0.32    0.000167          41         4           close
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         3           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         2           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0         4           openat
  0.00    0.000000           0         4           newfstatat
  0.00    0.000000           0         1           set_robust_list
  0.00    0.000000           0         1           prlimit64
  0.00    0.000000           0         1           getrandom
  0.00    0.000000           0         1           rseq
------ ----------- ----------- --------- --------- ----------------
100.00    0.052862           4     10872         2 total
```

Buffer_size: 4096

```
~/Desktop/homework/SystemProgramming/hw4 ❯ strace -c ./fileperf input.txt output.txt 4096
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0       142           read
  0.00    0.000000           0       140           write
  0.00    0.000000           0         4           close
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         3           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         2           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0         4           openat
  0.00    0.000000           0         4           newfstatat
  0.00    0.000000           0         1           set_robust_list
  0.00    0.000000           0         1           prlimit64
  0.00    0.000000           0         1           getrandom
  0.00    0.000000           0         1           rseq
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000           0       320         2 total
```

ltrace & strace

ltrace & strace 的說明(Google 找到的):

ltrace的功能是能够跟踪进程的库函数调用，它会显现出哪个库函数被调用，而strace则是跟踪程序的每个系统调用。ltrace与strace使用的技术大体相同，但ltrace在对支持fork和clone方面，不如strace。strace在收到frok和clone等系统调用后，做了相应的处理，而ltrace没有。

————————————————————

4. 有辦法根據 2 和 3 分析一下「呼叫作業系統核心函數（ system call ）」和 「函數庫呼叫」的「成本」差異嗎？

整理數據:

- ltrace: 三個的 fgetc 的呼叫次數都是 572463, fputs 都是 93804 次

buffer size	0	-1	4096
time(second)	35	23	22

- strace:

buffer size	0	-1	4096
call times(read function)	572464	10692	142
time(second)	1.71	0.05	0.0

- 結論

  system call 的 時間和呼叫次數 都會隨 buffer size 而改變， fucnction call 的呼叫次數並不影響

  而呼叫和時間的成本:

time / call times	0	-1	4096
fucnction call	35/572464 = 0.00006	23/572464	22/572464
system call	1.71/572464 = 0.000002	0.05/10692	0.0/142

  可見 system call 的成本較低

註: 有參考學長姐的作業