## 作業4

## 408210005 謝宗哲

1. 分別使用不同的 buffersize 執行:

```
aqua@aqua-ubuntu:~/system-programming/hw$ time ./fileperf ./input.txt ./testOut.txt 0
        0m13.858s
aqua@aqua-ubuntu:~/system-programming/hw$ time ./fileperf ./input.txt ./testOut.txt -1
        0m0.747s
sys 0m0.446s
<mark>aqua@aqua-ubuntu:~/system-programming/hw</mark>$ time ./fileperf ./input.txt ./test0ut.txt 4096
real
        0m0.240s
        0m0.208s
        0m0.023s
aqua@aqua-ubuntu:~/system-programming/hw$ time ./fileperf ./input.txt ./test0ut.txt 16384
real
        0m0.261s
        0m0.225s
aqua@aqua-ubuntu:~/system-programming/hw$ time ./fileperf ./input.txt ./test0ut.txt 65536
real
        0m0.033s
aqua@aqua-ubuntu:-/system-programming/hw$ time ./fileperf ./input.txt ./testOut.txt 1048576
        0m0.186s
        0m0.028s
aqua@aqua-ubuntu:~/system-programming/hw$ time ./fileperf ./input.txt ./testOut.txt 8388608
        0m0.185s
        0m0.156s
        0m0.029s
aqua@aqua-ubuntu:~/system-programming/hw$
```

可以看出當 buffersize = 0 (unbuffered )的時候,花費時間最多 再來是 buffersize = -1 (linebuffered),花費時間有明顯的減少 最後是 fully buffered,花費時間比上面兩者都少了許多,但是隨著 buffersize 成長,時間並沒有明顯降低的趨勢 2. 使用 Itrace 觀察呼叫函數庫的情況, 分別測試 unbuffered, linebuffered, buffersize = 4K 的情况:

		system-program	nming/hw	<pre>\$ ltrace -c ./fileper</pre>	f ./sourceFile.txt				
./testOu % time		usecs/call	calls	function					
36.87	0.004718	4718		fopen					
17.82	0.002281	2281		fputc					
11.38	0.001457	1457		fputs					
8.79	0.001125	1125		getc					
8.14	0.001042			setvbuf					
7.20	0.000922	922		memset					
6.52	0.000834	834		isoc99_sscanf					
3.27	0.000419	419	1.	fprintf					
100.00	0.012798	9	8	total					
		system-program			f /sourceFile txt				
<pre>aqua@aqua-ubuntu:~/system-programming/hw\$ ltrace -c ./fileperf ./sourceFile.txt ./testOut.txt -1</pre>									
% time		usecs/call	calls	function					
26.48	0.005239	5239	1	getc					
20.36	0.004029	4029	1	fopen					
16.10	0.003185	3185	1	fputs					
15.93	0.003151	3151	1	fputc					
7.64	0.001512	1512	1						
5.12	0.001014	1014		fprintf					
4.63	0.000916	916		memset					
3.74	0.000740	740	1	setvbuf					
100.00	0.019786			total					
		system-progra	mming/hw	<pre>\$ ltrace -c ./fileper</pre>	f ./sourceFile.txt				
	t.txt 4096								
% time	seconds	usecs/call	calls	function					
	0.000505								
27.68	0.000581	581		fopen					
11.82	0.000248	248		isoc99_sscanf					
11.67	0.000245	245		setvbuf					
10.96	0.000230	230		getc					
10.34	0.000217	217		memset					
9.29	0.000195	195	1	fprintf					

除了原本程式裡面呼叫的函數, 還有 fputs, fputc 這兩個沒寫在程式裡的函數, 可能是編譯器把 fprintf 轉成這兩個函數的。

1 fputc

8 total

9.15

9.10

0.000192

0.000191

100.00 0.002099

但是我不太理解為什麼每個函數的呼叫次數只有 1 次,像是 getc 在迴圈裡面每次都會讀一個字元,呼叫次數應該會跟檔案長度一樣。

3. 使用 strace 觀察呼叫作業系統的情況,分別測試 unbuffered, linebuffered, buffersize = 4K 的情況:

agua@agu	a-ubuntu:~/	system-progra	amming/hw\$	strace -c ./fileperf	./sourceFile.txt
	it.txt 0		3,	- 1, 1-1-per,	51
% time	seconds	usecs/call	calls	errors syscall	52
					5-5
64.00	0.006720	8	816	read	54
14.49	0.001521	12	123	write	55
5.14	0.000540	77	7	mmap	36
4.91	0.000516	86	6	pread64	57
3.65		ing 95	4	openat	58
3.50	0.000368	92	4	mprotect	59
1.35	0.000142	47	3	brk	60
1.14	0.000120	120	1	munmap	61
0.63		33	2	1 arch_prctl	62
0.60	0.000063	31	2	fstat	63
0.58	0.000061	30	2	close	64
0.00	0.000000	0	1	1 access	65 //
0.00	0.000000	one- 0	1	execve	- 66 i f
100.00	0.010500		972	2 total	67
aqua@aqu	ua-ubuntu:~/	system-progra	amming/hw\$	strace -c ./fileperf	./sourceFile.txt
	ıt.txt -1				54
% time	seconds	usecs/call	calls	errors syscall	55
20.10	0.000635	90	7	mmap	57
18.11	0.000572	47	12	write	58
15.48	0.000489	81	6	pread64	59
13.55	0.000428	107	4	mprotect	50
8.29	0.000262	65	4	openat	61
6.62	0.000209	52	4	fstat	62
4.87	0.000154	51	3	brk	63
4.15	0.000131	43	3	read	64
3.96	0.000125	125	1	munmap	65 //:
2.53	0.000080	one- 40	2 2	1 arch_prctl	66 17
2.34 0.00	0.000074	37 0	1	close	67
0.00	0.000000	0	1	1 access	68
0.00	0.00000		т	execve	69
100.00	0.003159		50	2 total	70
aqua@aqu	ua-ubuntu:~/	system-progra	amming/hw\$	strace -c ./fileperf	./sourceFile.txt
	ut.txt 4096				06 21
% time	seconds	usecs/call	calls	errors syscall	67
					98
37.03	0.000267	66	4	openat	69
20.67	0.000149	49	3	read	7.0
18.72	0.000135	33	4	fstat	71.
14.98	0.000108	108	1	write	7.2
8.60	0.000062	20	3	brk	73
0.00	0.000000	0	2	close	7.4
0.00	0.000000		7	mmap	7.5
0.00	0.000000		4	mprotect	7.6
0.00	0.000000	Θ	1	munmap	777
0.00	0.000000		6	pread64	78
0.00	0.000000	0	ī	1 access	79
0.00	0.000000	0	ī	execve	86
0.00	0.000000		2	1 arch prctl	81
100.00	0.000721		39	2 total	99

可以從上面三張圖觀察到 unbuffered 的時候 read 被呼叫 816 次, write 被呼叫 123 次。 linebuffered 的時候 read 被呼叫 3 次, write 被呼叫 12 次。 而 4K buffer 的時候 read 被呼叫 3 次, write 被呼叫 1 次。

所以有無 buffer 對呼叫作業系統的次數是有很大的影響的。

4. 從 2 跟 3 的測試中可以看出每個 function call 所花的時間比 system call 多,但是 system call 被呼叫的次數也比 function call 還多。 總共花的時間是 system call 的部分比較少。