

<前置作業>

先學習如何寫 makefile

% :萬用配對字元

\$(@) :目前的目標項目名稱

\$(*) :代表目前的相依性項目，但不含副檔名

\$(wildcard *.c)語法 獲取工作目錄下的所有的.c 檔案列表

\$(patsubst %.c, %.o, \$(dir))語法 patsubst 把\$(dir)中的變數符合字尾是.c 的全部替換成.o

%是這個專案要建立的檔案、%.c 是相依性的項目或檔案

\$(CC) \$(FLAGS) \$@.c -o \$@ 則為要產生這個項目所要執行的命令

我利用在 all: 加上 flock.db 以及 lockf.db 並比照上述的規則

flock.db、lockf.db 代表要建立的檔案，並利用 echo 指令在建立檔案時把 3500 放進去

在 clean: 加上 flock.db、lockf.db 代表執行 make clean 要刪除檔案

```
nash@SleepyCat: ~/Desktop/system-programming-hw x nash@SleepyCat: ~/Desktop/system-programming-hw x
1 SHELL = /bin/bash
2 CC = gcc
3 CFLAGS = -g -pthread
4 SRC = $(wildcard *.c)
5 EXE = $(patsubst %.c, %, $(SRC))
6
7 all: ${EXE} flock.db lockf.db
8
9 %: %.c
10     ${CC} ${CFLAGS} $@.c -o $@
11
12 clean:
13     rm ${EXE}
14     rm flock.db
15     rm lockf.db
16
17 flock.db:
18     echo "3500" > flock.db
19 lockf.db:
20     echo "3500" > lockf.db
```

<Code>

測試一

寫了 flock.c 和 lockf.c

flock.c 裡面長這樣，我是設定互斥的鎖，參數是 LOCK_EX，然後我開了四個視窗同時執行./flock 在這種狀態下，其他 process 沒有辦法去寫入正在被使用的 flock.db 檔案，如下圖，其他視窗的程式都停留在 flock-----?。直到第一個使用 flock.db 的程式執行結束才開始動作。

```
21
22     printf("flock-----?\n ");
23     if (flock(fd,LOCK_EX)<0){
24         perror("flock_error ");
25         exit(EXIT_FAILURE);
26     }
27     printf("flock success\n");
28
```

```
nash@SleepyCat: ~/De... x nash@SleepyCat: ~/De... x nash@SleepyCat: ~/De... x nash@SleepyCat: ~/De... x
count :315
Now at :1733757
count :316
Now at :1737575
count :317
Now at :1741394
count :318
Now at :1745214
```

```
nash@SleepyCat: ~/De... x nash@SleepyCat: ~/De... x nash@SleepyCat: ~/De... x nash@SleepyCat: ~/De... x
nash@SleepyCat:~/Desktop/system-programming-hw$ ./flock
flock-----?
█
```

我利用 time 指令來觀察執行./flock 的時間，發現 Realtime >>userCPU_Time + KernelCPU_Time 這感覺是蠻合理的事情，因為讀寫檔案時都在處理和外部硬碟的 I/O，時間都花在 I/O 上了，

```
real    1m41.046s
user    0m0.141s
sys     0m0.000s
nash@SleepyCat:~/Desktop/system-programming-hw$
```

測試三

檔案大小的話以 3500 OOOOOOO...OOO 估算的話，

平均我用 檔案大小/(空格+檔案大小)*3.9M

$4000/(3504+3505+3506+...+4504)=(4000/4008004)*3.9M = 3892 \text{ bytes(實際上的大小)}$

但這種寫程式法感覺會跨過很多 4k blocks 所以才會顯示 3.8M 吧。

```
nash@SleepyCat:~/Desktop/system-programming-hw$ ls -lhs
total 3.9M
 20K -rwxrwxr-x 1 nash nash 20K  29 18:55 flock
 4.0K -rw-rw-r-- 1 nash nash 1.1K 29 18:51 flock.c
 3.8M -rw-rw-r-- 1 nash nash 3.9M 29 18:57 flock.db
```

用 lockf 測試的話，一樣會互斥，這是取決於自己下的參數

```
nash@SleepyCat: ~/Desкто... x nash@SleepyCat: ~/Desкто... x nash@SleepyCat: ~/Desкто... x nash@SleepyCat: ~/Desкто... x
count :246
Now at :888375
count :247
Now at :892124
count :248
Now at :895874
count :249
Now at :899625

nash@SleepyCat: ~/Desкто... x nash@SleepyCat: ~/Desкто... x nash@SleepyCat: ~/Desкто... x nash@SleepyCat: ~/Desкто... x
Now at :66752
nash@SleepyCat:~/Desktop/system-programming-hw$ vim lockf.c
nash@SleepyCat:~/Desktop/system-programming-hw$ make clean
rm flock lockf
rm flock.db
rm lockf.db
nash@SleepyCat:~/Desktop/system-programming-hw$ ./lockf
lockf-----?
```

測試二

程式還是執行了 1000 次迴圈，但在 ./flock 或是 ./lockf 更新檔案時，vim 會跳出提醒，這裡我倒是不清楚測試二究竟要做出怎麼樣的結果才是合理的

```
@
@
@
"lockf.db"
WARNING: The file has been changed since reading it!!!
Do you really want to write to it (y/n)?
```

參考資料

簡單學 makefile : makefile 介紹與範例程式

<https://mropengate.blogspot.com/2018/01/makefile.html>

makefile 中的 patsubst

<https://www.itread01.com/content/1545879139.html>

time (LINUX 系統命令)

<https://baike.baidu.com/item/time/13023979>

致謝

我自己

想說的話

幹要爆炸了