System Programming Homework 02 資工2B 408410120 鍾博丞

環境配置

Operating System: Ubuntu 20.04 LTS using KDE plasma

CPU: AMD R9 3900X 12C 24T @ 3.8GHz

SSD: WD Black 256G WDS256G1X0C TLC (Seq. R: 2050MB/s, Seq. W: 700MB/s, Random R: 170K IOPS, Random W: 130K IOPS)

執行環境

使用者只需下達 `make run` 就可以了；刪除檔案時如果要連帶 FileHole 等測試檔案一起刪除，只需下達 `make clean_all` 就可以了

In makefile

```
SHELL = /bin/bash
CC = gcc
CFLAGS = -g -pthread
SRC = $(wildcard *.c)
EXE = $(patsubst %.c, %, $(SRC))

all: ${EXE}

%:  %.c
    ${CC} ${CFLAGS} $@.c -o $@

run: all
    ./hole
    @printf '\n'
    time ./mycp2 FileHole FileHole2
    @printf '\n'
    time ./mmap_cp2 FileHole FileHole3
    @printf '\n'
    time ./mmap_cp FileHole FileHole4
    @printf '\n'
    ls -lash FileHole FileHole2 FileHole3 FileHole4
    @printf '\n'

clean:
    rm ${EXE}

clean_all:
    rm ${EXE} FileHole FileHole2 FileHole3 FileHole4
```

我有修改 hole.c 使它產生的檔案大一點

In hole.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <assert.h>
#include <time.h>

int main() {
    ssize_t fd;
    const off_t G = (1 << 30);
    const off_t M = (1 << 20);

    printf("這個程式會在當前的目錄下，製造檔案FileHole\n");
    fd = open("./FileHole", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    if (fd < 0)
        perror("無法製造myHole");
    //ftruncate(fd, 6*G);
    //lseek將『檔案指標』由開始位置向後移動 1G，lseek比較可能出錯，用 assert 檢查一下

    time_t timer1, timer2;
    timer1 = time(NULL);

    assert(lseek(fd, G - 200 * M, SEEK_SET) != -1);
    //寫入“1”，很少出錯，懶得檢查
    for (off_t i = 0; i < 200 * M; ++i) {
        write(fd, "1", sizeof("1"));
    }
    //lseek將『檔案指標』由『目前』位置向後移動 2G，lseek比較可能出錯，用 assert 檢查一下
    assert(lseek(fd, 2 * G - 100 * M, SEEK_CUR) != -1);
    for (off_t i = 0; i < 100 * M; ++i) {
        write(fd, "2", sizeof("2"));
    }
    //lseek將『檔案指標』由『目前』位置向後移動 3G，lseek比較可能出錯，用 assert 檢查一下
    assert(lseek(fd, 3 * G - 200 * M, SEEK_CUR) != -1);
    for (off_t i = 0; i < 200 * M; ++i) {
        write(fd, "3", sizeof("3"));
    }

    timer2 = time(NULL);

    printf("time(create FileHole) = %ld sec \n", timer2 - timer1);

    close(fd);
    system("ls FileHole -alhs");
    return 0;
}
```

測試結果如下

```
make run
gcc -g -pthread mmap_cp.c -o mmap_cp
gcc -g -pthread hole.c -o hole
gcc -g -pthread mycp2.c -o mycp2
gcc -g -pthread mmap_cp2.c -o mmap_cp2
./hole
這個程式會在當前的目錄下，製造檔案FileHole
time(create FileHole) = 6 sec
1006M -rw------- 1 ubuntu2004 ubuntu2004 6.5G  3月 13 15:17 FileHole
```

```
time ./mycp2 FileHole FileHole2

real    0m0.672s
user    0m0.012s
sys     0m0.660s

time ./mmap_cp2 FileHole FileHole3

File size = 6971981824          Bytes
File size = 6808576.0000        KB
File size = 6649.0000           MB
File size = 6.0000              GB

mmap: Success
inputPtr = 0x7fe243616000
mmap, output: Success
outputPtr = 0x7fe0a3d16000

real    0m0.557s
user    0m0.065s
sys     0m0.412s

time ./mmap_cp FileHole FileHole4
file size = 6971981824
mmap: Success
inputPtr = 0x7f2d3a992000
mmap, output: Success
outputPtr = 0x7f2b9b092000
memory copy
time(memcpy) = 23 sec

real    0m22.779s
user    0m0.306s
sys     0m4.778s

ls -lash FileHole FileHole2 FileHole3 FileHole4
1006M -rw------- 1 ubuntu2004 ubuntu2004 6.5G  3月 13 15:17 FileHole
1006M -rw------- 1 ubuntu2004 ubuntu2004 6.5G  3月 13 15:17 FileHole2
1006M -rw------- 1 ubuntu2004 ubuntu2004 6.5G  3月 13 15:17 FileHole3
 6.5G -rw------- 1 ubuntu2004 ubuntu2004 6.5G  3月 13 15:17 FileHole4
```

從這裡可以明顯看出，mycp2 用了0.66s 做 I/O；本次作業實做的 mmap_cp2 用了 0.412s 做 I/O。

而不跳過 file hole 的 mmap_cp 更是總共用了 22.779s 才完成

由本次實驗得到，mmap 的方式複製較 RW 為快速

最後的壓縮指令

`tar jcvf filename.tar.bz2 target`