High

Performance

Distributed

System

KUAS – High Performance Distributed System

Working with Files

HPDS

Low-level file access

- 0 : standard input
- 1 : standard output
- 2 : standard error

HPDS

Low-level file access

- open
- read
- write
- close
- ioctl

open

- int open(const char* path, int oflags);
- int open(const char* path, int oflags, mode_t mode);
- Mode
  - O_RDONLY
  - O_WRONLY
  - O_RDWR
- Oflags:
  - O_APPEND
  - O_TRUNC (放棄現有的內容、長度歸零)
  - O_CREAT
  - O_EXCL （確保呼叫者可以產生檔案）

mode

- S_IRUSR
- S_IWUSR
- S_IXUSR
- S_IRGRP
- S_IWGRP
- S_IWGRP
- S_IROTH
- S_IWOTH
- S_IXOTH

read

- #include <unistd.h>
- size_t read(int fildes, void* buf, size_t nbytes);

read

```
#include <unistd.h>

int main()
{
    char buffer[128];
    int nread;

    nread = read(0, buffer, 128);
    if (nread == -1)
        write(2, "A read error has occurred\n", 26);

    if ((write(1,buffer,nread)) != nread)
        write(2, "A write error has occurred\n",27);

    exit(0);
}
```

write

- #include <unistd.h>
- size_t write(int fildes, void* buf, size_t n bytes);

write

```
#include <unistd.h>

int main()
{
    if ((write(1, "Here is some data\n", 18)) != 18
)
        write(2, "A write error has occurred on file
descriptor 1\n",46);

    exit(0);
}
```

File copy

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

int main()
{
    char c;
    int in, out;

    in = open("file.in", O_RDONLY);
    out = open("file.out", O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);
    while(read(in,&c,1) == 1)
        write(out,&c,1);

    exit(0);
}
```

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

int main()
{
    char block[1024];
    int in, out;
    int nread;

    in = open("file.in", O_RDONLY);
    out = open("file.out", O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);
    while((nread = read(in,block,sizeof(block))) > 0)
        write(out,block,nread);

    exit(0);
}
```

HPDS

Standard I/O Library

- fopen, fclose
- fread, fwrite
- fflush
- fseek
- fgetc, getc, getchar
- fputc, putc, putchar
- fgets, gets
- printf, fprintf, sprintf
- scanf, fscanf, sscanf

fopen

- #include <stdio.h>
- FILE *fopen(const char *filename, const char* mode);
- “r” （唯讀模式）
- “w” （寫入模式、長度歸零）
- “a” （寫入模式、附加）

- size_t fread(void* ptr, size_t size, size_t n items, FILE* stream);
- size_t fwrite(void* ptr, size_t size, size_t nitems, FILE* stream);
- int fclose(FILE* stream);
- int fflush(FLIE* stream);
- int fseek(FILE* stream, long int offset, in t whence);

- int fgetc(FILE* stream);
- int getc(FILE* stream);
- int getchar();
- int fputc(int c, FILE* stream);
- int putc(int c, FILE* stream);
- int putchar(int c);
- char* fgets(char*s, int n, FILE* stream);
- char* gets(char*s);

printf, sprintf, fprintf,

- %d
- %o, %x
- %c
- %s
- %f （float)
- %e (double)
- %g (double)

scanf, fscanf, sscanf

- %d
- %o, %x
- %f, %e, %g
- %c,
- %s
- %[]　（掃瞄特定字元）
- %%　（掃瞄 % 的字元）

```
#include <stdio.h>

int main()
{
    int c;
    FILE *in, *out;

    in = fopen("file.in","r");
    out = fopen("file.out","w");

    while((c = fgetc(in)) != EOF)
        fputc(c,out);

    exit(0);
}
```

- #include <sys/stat.h>
- int mkdir(const char* path, mode_t mode);
- int rmdir(const char* path);

- #include <unistd.h>
- int chdir(const char *path);
- char* getcwd(char* buf, size_t size);

- #include <sys/types.h>
- #include <dirent.h>

- DIR* opendir(const char* name);
- struct dirent *readdir(DIR *dirp);
- long int telldir(DIR* dirp);
- void seekdir(DIR* dirp, long int loc);
- int closedir(DIR* dirp);

```
void printdir(char *dir, int depth)
{
    DIR *dp;
    struct dirent *entry;
    struct stat statbuf;

    if((dp = opendir(dir)) == NULL) {
        fprintf(stderr,"cannot open directory: %s\n", dir);
        return;
    }
    chdir(dir);
    while((entry = readdir(dp)) != NULL) {
        lstat(entry->d_name,&statbuf);
        if(S_ISDIR(statbuf.st_mode)) {
            /* Found a directory, but ignore . and .. */
            if(strcmp(".",entry->d_name) == 0 ||
                strcmp("..",entry->d_name) == 0)
                continue;
            printf("%*s%s/\n",depth,"",entry->d_name);
            /* Recurse at a new indent level */
            printdir(entry->d_name,depth+4);
        }
        else printf("%*s%s\n",depth,"",entry->d_name);
    }
    chdir("..");
    closedir(dp);
}
```

```
int main()
{
    printf("Directory scan of /home:\n");
    printdir("/home",0);
    printf("done.\n");


    exit(0);
}
```

mmap

- void *mmap(void *addr, size_t len, int port, int flags, int fildes, off_t off);
- PORT_READ
- PORT_WRITE
- PORT_EXEC
- PORT_NONE
- MAP_PRIVATE
- MAP_SHARED
- MAP_FIXED

msync

- int msync(void* addr, size_t len, int flags);
- MS_AYSNC （非同步寫入）
- MS_SYNC （同步寫入）
- MS_INVALIDATE （再從檔案讀）

munmap

- int munmap(void * addr, size_t len);

```
typedef struct {
    int integer;
    char string[24];
} RECORD;

#define NRECORDS (100)

int main()
{
    RECORD record, *mapped;
    int i, f;
    FILE *fp;

    fp = fopen("records.dat","w+");
    for(i=0; i<NRECORDS; i++) {
        record.integer = i;
        sprintf(record.string,"RECORD-%d",i);
        fwrite(&record,sizeof(record),1,fp);
    }
    fclose(fp);
```

```
/*  We now change the integer value of record 43 to 143
    and write this to the 43rd record's string.  */

    fp = fopen("records.dat","r+");
    fseek(fp,43*sizeof(record),SEEK_SET);
    fread(&record,sizeof(record),1,fp);

    record.integer = 143;
    sprintf(record.string,"RECORD-%d",record.integer);

    fseek(fp,43*sizeof(record),SEEK_SET);
    fwrite(&record,sizeof(record),1,fp);
    fclose(fp);
```

```
/*  We now map the records into memory
    and access the 43rd record in order to change the integer to 243
    (and update the record string), again using memory mapping.  */

    f = open("records.dat",O_RDWR);
    mapped = (RECORD *)mmap(0, NRECORDS*sizeof(record),
                        PROT_READ|PROT_WRITE, MAP_SHARED, f, 0);

    mapped[43].integer = 243;
    sprintf(mapped[43].string,"RECORD-%d",mapped[43].integer);

    msync((void *)mapped, NRECORDS*sizeof(record), MS_ASYNC);
    munmap((void *)mapped, NRECORDS*sizeof(record));
    close(f);

    exit(0);
}
```

```
int main(int argc, char* argv[])
{
    char *topdir, pwd[2]=".";
    if (argc != 2)
        topdir=pwd;
    else
        topdir=argv[1];

    printf("Directory scan of %s\n",topdir);
    printdir(topdir,0);
    printf("done.\n");

    exit(0);
}
```

練習一：

- 產生 10 個亂數，用 fprintf 寫入至 data in
- 用 fscanf 從 datain 讀出 10 個亂數，做排序

  （由小到大）。
- 把排序結果 fprintf 寫到 dataout

練習二

- 宣告一個結構 point: float x, float y, float dist 。
- 宣告一個可容納 10 個結構的陣列 A 。
- 用亂數初始十個點座標。
- 用 fwrite 寫入檔案 points 。
- 再用 fread 讀出，並比較其結果。

練習三

- 用 mmap 將檔案 points 對映至 point 結構矩陣 B 。
- 計算每個座標點至原點的距離，並寫入矩陣 B 裡每個座標點的 dist 欄位。
- 用 msysc 將結果寫回檔案 points
- 再用 fread 將十個座標點讀出，並檢查其至原點距離是否正確？