บทที่ 3

ฟังก์ชันและโปรแกรมที่เกี่ยวข้อง

ฟังก์ชันและโปรแกรมที่เกี่ยวข้อง มีอยู่หลายกลุ่มด้วยกัน คือ

1. CURSES เป็นฟังก์ชันสำเร็จรูปที่มีอยู่ในระบบปฏิบัติการ โดยที่เป็นฟังก์ชันที่จัดการเกี่ยวกับจอภาพ รวมถึงฟังก์ชันที่มีการรับค่าจาก keyboard โดยไม่ต้องกดปุ่ม return (หรือ enter) จึงทำให้สะดวกในการใช้งาน โดยในโปรแกรมมีการใช้ฟังก์ชันของ CURSES อยู่หลายฟังก์ชันคือ

(1) ฟังก์ชัน initscr

ฟังก์ชันนี้จะต้องใช้ก่อนที่จะมีการใช้ฟังก์ชันใน CURSES โดยที่ฟังก์ชันนี้เป็นการ initialize จอภาพให้สามารถใช้ฟังก์ชันใน CURSES ได้นั่นเอง

(2) ฟังก์ชัน endwin

ฟังก์ชันนี้ เป็นฟังก์ชันที่จะใช้ก็ต่อเมื่อไม่มีการใช้ฟังก์ชันของ CURSES แล้ว กล่าวคือ จะใช้เป็นฟังก์ชันสุดท้ายของ CURSES ก่อนที่จะจบการทำงานของโปรแกรม

(3) ฟังก์ชัน getch

ฟังก์ชันนี้ เป็นฟังก์ชันในการรับค่าตัวอักษร 1 ตัว เข้ามาในโปรแกรม โดยไม่ต้องรอการกด return อีกทั้งฟังก์ชันนี้ยังมีประโยชน์ในการจัดาการเกี่ยวกับปุ่มลูกศร ขึ้น, ลง, ซ้าย, ขวา ได้อีกด้วย

(4) ฟังก์ชัน refresh

เป็นการนำผลของคำสั่งที่จัดการเกี่ยวกับจอภาพให้แสดงออกทางหน้าจอ เพราะเมื่อมีการใช้คำสั่งที่จัดการเกี่ยวกับจอภาพ ภายใต้การทำงานของ CURSES โปรแกรมจะไม่แสดงผลทันที จนกว่าจะมีการสั่ง refresh ขอบเขตของการใช้คำสั่งเกี่ยวกับจอภาพ ยังครอบคลุมถึงการใช้คำสั่งในลักษณะ escape sequence บนจอ VT ในบางครั้งอีกด้วย

ฟังก์ชันใน CURSES ยังมีฟังก์ชันที่น่าสนใจอีกมาก แต่เนื่องจากยังไม่มีความจำเป็นในการใช้ อีกทั้งยังสามารถใช้คำสั่งในลักษณะการส่ง escape sequece ออกทางจอภาพแทนได้ (ซึ่งจะกล่าวในหัวข้อต่อไป) จึงไม่ขอกล่าวถึงฟังก์ชันของ CURSES ที่เหลือ ส่วนวิธีการใช้ฟังก์ชันของ CURSES นี้ ที่ส่วนบนของ sort program จะต้องใส่ include < curses.h > ในทุก ๆ program ที่มีการใช้ฟังก์ชันของ CURSES และเวลา compile จะต้องทำการ link รวมกับ library ของ CURSES ซึ่งมี 2 library คือ CURSES และ termcap เช่น file ชื่อ a.c จะใช้ฟังก์ชันใน cc a.c.lcurses - /termcap อีกทั้งก่อนที่จะมีการใช้ฟังก์ชันใน curses จะต้องมีการ set environment ของระบบ ให้เป็น VT 100 โดย

$ set term = vt100

2. คำสั่ง macro ในการส่ง escape sequence ในการจัดการเกี่ยวกับจอภาพ ซึ่งจะ define ค่า macro ที่จัดการกับจอภาพทั้งหมด อยู่ใน file ชื่อ scr.h แต่ในที่นี้จะกล่าวถึงเฉพาะที่ใช้งานเท่านั้น ซึ่งจะมีรายละเอียดดังต่อไปนี้

(1) คำสั่ง dubln

เป็นคำสั่งที่กำหนดให้ ถ้าลักษณะที่พิมพ์ออกทางจอภาพในบรรทัดนั้น มีขนาดความกว้างเป็นสองเท่าของตัวอักษรปกติ นั่นคือตำแหน่ง cursor อยู่ที่บรรทัดใด แล้วใช้คำสั่ง dubln บรรทัดนี้จะมีความกว้างตัวอักษรเป็นสองเท่าของบรรทัดที่มีตัวอักษรปกติ

(2) คำสั่ง clreol

เป็นคำสั่งที่ใช้ลบตัวอักษรบนจอภาพตั้งแต่ตำแหน่งที่ cursor อยู่ไปจนกระทั่งตัวอักษรตัวสุดท้ายของบรรทัด

(3) คำสั่ง clrscr

เป็นคำสั่งที่ใช้ในการลบจอภาพ

(4) คำสั่ง scr132

เป็นคำสั่งที่ใช้ในการ set จอภาพให้มีขนาด 132 column ต่อบรรทัด

(5) คำสั่ง scr80

เป็นคำสั่งที่ใช้ในการ set จอภาพให้มีขนาด 80 column ต่อบรรทัด

(6) คำสั่ง norchr

เป็นคำสั่งที่ใช้ในการ set ตัวอักษรที่แสดงผลบนจอภาพ ให้มี attribute เป็น normal video (ตัวปกติ)

(7) คำสั่ง hghchr

เป็นคำสั่งที่ใช้ในการ set ตัวอักษรที่แสดงผลบนจอภาพให้มี attribute เป็น high-light video (ตัวเข้ม)

(8) คำสั่ง blkchr

เป็นคำสั่งที่ใช้ในการ set ตัวอักษร ที่แสดงผลบนจอภาพให้มี attribute เป็น blinking video (ตัวกะพริบ)

(9) คำสั่ง revchr

เป็นคำสั่งที่ใช้ในการ set ตัวอักษรที่แสดงผลบนจอภาพให้มี attribute เป็น reverse video (สีพื้นกับสีตัวอักษรจะกลับกัน)

(10) คำสั่ง scrpos

เป็นคำสั่งที่ใช้ในการเคลื่อนย้ายตำแหน่งของ cursor ไปบนจอภาพตามตำแหน่งที่ต้องการ โดยเราจะต้องให้ค่า line และ column แก่คำสั่ง

ในการที่จะใช้คำสั่ง macro เหล่านี้ที่ส่วนบนของ sort program ภาษา C จะต้องเพิ่ม # include "scrsh"

3. function ในการจัดการ indexed file ชนิด inx function นี้ เขียนโดยโปรแกรมเมอร์ของศูนย์คอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ โดยจะมีทั้งในส่วนของคำสั่งที่เรียกใช้ได้โดยตรง บน shell และ function ที่ต้องการ library และทำการ link รวมกับโปรแกรมที่เขียนขึ้น ด้วยภาษา C แต่ในส่วนของโครงงานนี้จะใช้ในลักษณะของ function ภาษา C เท่านั้น โดยมีรายละเอียดดังต่อไปนี้

ชื่อ

fopeninx - เปิด indexed file

รูปแบบ

```
ret = fopeninx (file_name, type, fp)
int ret
char *file_name, *type;
FILE *fp[2];
```

คำอธิบาย

fopeninx เปิด indexed file โดยมี 2 file คือ data file กับ inx file (exam: "name" and "name.inx") ซึ่งไฟล์ทั้ง 2 นี้มีโครงสร้างเหมือนกับ indexed file ของ rmcobol

file_name เป็น string เก็บชื่อ file

fp เป็น pointer สำหรับเปิด data file กับ inx file

fp[0] สำหรับ data file

fp[1] สำหรับ inx file

type เก็บเงื่อนไขการเปิด file ซึ่งมีค่าดังนี้

"r" เปิดเพื่อ read

"r+" เปิดเพื่อ read, update, append

ret = NULL หมายถึงเปิด file ไม่สำเร็จ

ชื่อ

fcreatinx - สร้าง indexed file

รูปแบบ

ret = fcreatinx (file_name, rec_len, key_offset, fp)

int ret, rec_len, key_len, key_offset;

char file_name[80];

FILE *fp[2]

คำอธิบาย

fcreatinx - สร้าง indexed file

file_name เป็น string เก็บชื่อไฟล์

rec_len เป็นความยาวของ record ของ indexed file

key_len เป็นความยาวของ key ของ indexed file

key_offset เป็นตำแหน่งเริ่มต้นของ key

เช่น key_offset = 0 หมายถึง key เริ่มคอลัมน์ 1

เมื่อใช้ fcreatinx แล้วไฟล์จะเปิดให้ใช้ read, write ได้เลย

ret = NULL หมายถึงเปิด file ไม่สำเร็จ

ชื่อ

freadinx - read indexed file

fwriteinx - append indexed file

frewriteinx - update indexed file

รูปแบบ

In = freadinx (buf, key, fp);

long ln;

char buf[BIFSIZ], key[200];

FILE *fp[2];

ret = fwriteinx (buf, fp, mode);

int ret;

char buf[BIFSIZ], mode[2];

FILE *fp[2];

ret = frewriteinx (buf, ln, fp);

long ln;

char buf[BIFSIZ];

FILE *FP[2]

คำอธิบาย

freadinx read indexed file โดยใช้ key เป็นคีย์ในการอ่านข้อมูลที่อ่านได้เก็บใน buf

ln เป็น record no. ของข้อมูลที่อ่านได้

ln = OL หมายถึง NEW record

fwriteinx append index file โดย write ข้อมูลจาก buf ต่อท้าย data file

-ret > .0 หมายถึง write สำเร็จ

ret = -1 หมายถึง duplicate key, write ไม่สำเร็จ

mode != "m" ไม่มีการตรวจสอบก่อน write

frewriteinx update indexed file โดย write ข้อมูลจาก buf ต่อท้าย data file.

ret = 1 หมายถึง rewrite สำเร็จ

ret = -1 หมายถึง rewrite ไม่สำเร็จ

fcloseinx - ปิด indexed file

รูปแบบ

fcloseinx (fp);

FILE *fp[2];

คำอธิบาย

fcloseinx - ปิด indexed file คือไฟล์ data file และ inx file

ส่วนวิธีการใช้คำสั่งบน shell จะดูเพิ่มเติมได้จากคำสั่ง helpinx บน shell โดยใช้คำสั่ง

sritrang > /staff/mongkol/bin/helpinx

และในการใช้ function ที่จัดการกับ indexed file กลุ่มนี้ (กลุ่ม "inx") สามารถที่จะใช้ร่วมกับ indexed file ของ rmcobol ได้ กล่าวคือ function เหล่านี้ใช้ได้กับ indexed file ที่จัดการโดย rmcobol และ rmcobol ก็สามารถใช้ indexed file ที่จัดการโดย function เหล่านี้

ส่วน library ของ function ที่จัดการเกี่ยวกับ indexed file เหล่านี้ คือ

/staff/mongkol/bin/libinx

ซึ่งจะต้องนำไป link รวมกับ object ของ file อื่น ๆ อีก

4. คำสั่ง make เป็นคำสั่งในระบบปฏิบัติการ ลักษณะของคำสั่ง make คือ คำสั่ง make จะใช้ข้อมูลใน Makefile หรือ makefile (งานวิจัยนี้ใช้ Makefile) เป็นแนวทางในการทำงานของคำสั่ง ซึ่ง Makefile นี้จะบอกถึงโปรแกรมย่อยต่าง ๆ ที่จะนำมารวมเป็นโปรแกรมใหญ่ ลักษณะการ compile คำสั่ง make จะตรวจสอบว่าโปรแกรมย่อยได้ compile เป็น object file หรือยัง หรือโปรแกรมย่อยได้มีการแก้ไขหรือไม่ถึงจะทำการ compile ให้ในกรณีที่ยังไม่ compile โปรแกรมย่อยเป็น object file หรือมีการแก้ไขโปรแกรมย่อย ซึ่งถ้าเกิดกรณีนี้ขึ้นก็จะ compile โปรแกรมย่อยแล้วนำไป link กับโปรแกรมย่อยอื่น ๆ ให้เป็น executable file ต่อไป

ตัวอย่าง ใน file ชื่อ Makefile

```
out  :  a.0  c.0
        cc  a.0  b.0  c.0  -0  out
a.0  :  a.c
        cc - c  a.c
b.0  :  b.c
        cc - c  b.c
c.c  :  c.c
        cc - c  c.c
```

โดยที่ cc คือคำสั่งของ c compile และ parameter

- c หมายถึงให้ compile เป็น object file

- 0 หมายถึงหลังจาก compile แล้วให้เอา file ผลลัพธ์ไปไว้ในชื่อที่อยู่หลัง "-0" หลังจากที่ใช้คำสั่ง make ในครั้งแรกที่ยังไม่มี file ใด ทำเป็น object file คือ

```
sritrang > make
```

คำสั่ง make ก็จะทำงานตามลำดับดังต่อไปนี้

```
cc  -c  a.c
cc  -c  b.c
cc  -c  c.c
cc  a.0  b.0  c.0  -0  out
```

เมื่อสิ้นสุดการทำงาน ก็จะได้ file ชื่อ a.0, b.0, c.0 และ out จากนั้นจะสมมุติว่ามีการผิดพลาดในส่วนของ b.c เมื่อแก้ไขแล้วใช้คำสั่ง make อีกครั้ง โปรแกรม make จะทำเพียง

```
cc  -c  b.c
cc  a.0  b.0  c.0  -0  out
```

จึงประหยัดเวลาในการ compile มากขึ้น เพราะไม่ต้อง compile ใหม่ทั้งหมดโดยเฉพาะในกรณีที่มี file มาก ๆ