บทที่ 2

ชุดคำสั่งเคิร์ส

2.1 ความเป็นมา

เคิร์สเป็นชุดคำสั่ง (library) ที่ใช้ในการควบคุมจอภาพ (screen) ชุดคำสั่งเคิร์สนี้ทำให้โปรแกรมเมอร์สามารถควบคุมจอภาพโดยใช้ โครงสร้างข้อมูลที่เรียกว่า วินโดว์ (window) ซึ่งจะไม่ขึ้นกับลักษณะของเทอร์มินอล (terminal) ในการใช้ชุดคำสั่งเคิร์สนี้ทำให้โปรแกรมเมอร์ไม่จำเป็นต้องรู้รายละเอียดปลีกย่อยของเทอร์มินอลแต่ละประเภท เมื่อใดที่โปรแกรมเรียกใช้ชุดคำสั่งในเคิร์ส เคิร์สจะพยายามที่จะเปลี่ยนแปลงหน้าจอภาพให้มีประสิทธิภาพที่สุด ดังนั้นเคิร์สจึงมาจากคำว่า Cursor Motion Optimization

เคิร์สถูกพัฒนาขึ้นที่มหาวิทยาลัยแคลิฟอร์เนียเบิร์คเลย์ และเป็นส่วนหนึ่งของยูนิกซ์ (UNIX) ที่เรียกว่า Berkeley Software Distribution (BSD) หลังจากนั้นบริษัทเอทีแอนด์ที (AT&T) ได้นำเคิร์สเข้ามาเป็นส่วนหนึ่งของยูนิกซ์ของตน ที่เรียกว่า UNIX System V โดยได้ปรับปรุงให้มีความสามารถมากขึ้น

2.2 โครงสร้างของวินโดว์

วินโดว์เป็นลักษณะของการแสดงข้อมูลซึ่งกำหนดไว้ว่า ลักษณะรูปทรงสี่เหลี่ยมที่แสดงบนจอภาพเป็นอย่างไร ส่วนเคิร์สวินโดว์จะไม่ใช่ลักษณะทางกายภาพ แต่จะเป็นโครงสร้างข้อมูลภายใน ในขณะที่โปรแกรมเมอร์จัดการกับเคิร์สวินโดว์นั้น จะไม่มีอะไรเกิดขึ้นกับหน้าจอกายภาพจนกว่าจะเรียกใช้ฟังก์ชัน (function) wrefresh() ซึ่งหลังจากเรียกใช้ฟังก์ชันนี้แล้ว หน้าจอกายภาพจึงจะเปลี่ยนแปลง

เคิร์สได้กำหนดลักษณะ โครงสร้างข้อมูลของวินโดว์ไว้ใน แฟ้ม (file) ชื่อว่า /usr/include/curses.h ไว้ดังนี้

```
struct _win_st;
typedef struct _win_st WINDOW;
```

ในแฟ้ม /usr/include/curses.h หรือเขียนตามลักษณะข้อตกลงบนยูนิกซ์ว่า <curses.h> ได้ให้โครงสร้างข้อมูลของวินโดว์ 2 ตัว คือ stdscr และ curscr

วินโดว์ curscr จะเก็บข้อมูลที่แสดงสถานะว่า ขณะนี้หน้าจอกายภาพกำลังแสดงอะไรและวินโดว์ stdscr เป็นวินโดว์ที่ให้โปรแกรมเมอร์ใช้ หลังจากที่โปรแกรมเมอร์ได้จัดการให้วินโดว์ stdscr ให้มีลักษณะตามที่ต้องการแล้ว ก็จะต้องเรียก ฟังก์ชัน wrefresh() เพื่อทำการปรับหน้าจอกายภาพให้มีลักษณะตามวินโดว์ stdscr

2.3 โครงสร้างของโปรแกรมเคิร์ส

โดยทั่วไปโปรแกรมเคิร์ส จะมีลักษณะดังนี้

```
#include <curses.h>
main()
{
        initscr();
      /* main body */
        endwin();
        exit (0);
}
```

ในการเขียนโปรแกรมเคิร์สใดๆก็ตาม จะต้องใส่เฮดเดอร์ (header file) ชื่อ curses.h เสมอ ซึ่งในยูนิกซ์ซิสเตมไฟว์ จะรวมแฟ้ม stdio.h และแฟ้ม termio.h เข้ามาด้วย ในแฟ้ม curses.h นี้ จะประกอบด้วย โครงสร้างข้อมูลของวินโดว์ ตลอดจนตัวแปรและตัวกำหนดต่างๆ ซึ่งจำเป็นต้องใช้ในโปรแกรมเคิร์ส

ก่อนที่จะเรียกฟังก์ชันเคิร์สใดๆ ก็ตาม โปรแกรมเมอร์จำเป็นต้องเรียก ฟังก์ชัน initscr() เสมอ โดยฟังก์ชัน initscr() นี้จะทำหน้าที่เตรียมการ (initialize) โครงสร้างข้อมูลของเคิร์ส และกำหนดประเภทของเทอร์มินอลที่กำลังใช้ นอกจากนี้ initscr() จะทำหน้าที่เตรียมวินโดว์ curscr และ stdscr ให้โปรแกรมเมอร์ใช้ ขั้นตอนนี้เป็นการเตรียมการ ให้โปรแกรมอยู่ในสภาวะที่เรียกว่า in-curses mode

จากนั้นโปรแกรมจะเข้าสู่ขั้นตอนการทำงาน โดยใช้ชุดคำสั่งของเคิร์สเข้ามาทำงาน เช่น การสร้างหรือลบวินโดว์ , การสร้างกรอบล้อมรอบวินโดว์ , การนำวินโดว์ 2 ตัวมาซ้อนทับกัน หรือการพิมพ์ข้อความออกทางวินโดว์ที่ต้องการ

หลังจากที่ประมวลผลสำเร็จเสร็จแล้ว เทอร์มินอลจะต้องกลับเข้าสู่สภาวะปกติ ซึ่งคือสภาวะก่อนที่จะเข้าโปรแกรมเคิร์ส ซึ่งจะต้องเรียกฟังก์ชัน endwin() หลังจากนั้นก็ออกจากโปรแกรม

2.4 ฟังก์ชันที่เกี่ยวข้องกับการกำหนดสภาวะของโปรแกรม

หลังจากที่โปรแกรมเคิร์สเรียกฟังก์ชัน initscr() แล้ว โปรแกรมจะต้องทำการกำหนดสภาวะของเทอร์มินอล ให้พร้อมที่จะรับข้อมูลนำเข้า (input) และนำออก (output) โดยมีฟังก์ชันต่างๆ ดังนี้

keypad(win,bf) ถ้า bf เป็น true ฟังก์ชันนี้จะทำให้สามารถใช้แผงแป้นพิเศษ (keypad) ได้ ซึ่งถ้า bf เป็น false หรือไม่ได้กำหนดเอาไว้ โปรแกรมจะเรียกใช้แผงแป้นพิเศษไม่ได้

savetty() เป็นฟังก์ชันที่เก็บสถานะของเทอร์มินอลขณะนั้น ไว้ในบัฟเฟอร์ภายใน (internal buffer) ฟังก์ชันนี้จะถูกเรียกโดยอัตโนมัติ จากฟังก์ชัน initscr()

resetty() เป็นฟังก์ชันที่นำสถานะของเทอร์มินอล ที่เก็บไว้ในบัฟเฟอร์ภายในจากการเรียกฟังก์ชัน savetty() กลับเข้าสู่สภาวะเดิม

reset_shell_mode() เป็นฟังก์ชันที่ปรับสภาวะของเทอร์มินอล กลับเข้าสู่สภาวะก่อนที่จะเข้าโปรแกรมเคิร์ส หรือเรียก initscr()

reset_prog_mode() เป็นฟังก์ชันที่ตั้งสถานะของเทอร์มินอล ให้กลับคืนสู่สภาวะโปรแกรมเคิร์ส

2.5 ฟังก์ชันที่เกี่ยวข้องกับวินโดว์

```
WINDOW          *win;
int             nline,ncol,begy,begx;
win    =        newwin(nline,ncol,begy,begx);
```

ฟังก์ชันนี้จะสร้างวินโดว์เคิร์สที่มีจำนวนบรรทัดเป็น nline จำนวนคอลัมน์เป็น ncol โดยมีตำแหน่งซ้ายบนของวินโดว์ win ที่ y = begy และ x = begx

```
WINDOW          *win1,*win2;
int             overwrite(win1,win2);
```

ฟังก์ชันนี้จะนำข้อมูลของวินโดว์ใน win1 ทับข้อมูลของวินโดว์ win2

```
WINDOW          *orig,*sub;
int             nline,ncol,begy,begx;
sub    =        subwin(orig,nline,ncol,begy,begx);
```

ฟังก์ชันนี้จะสร้างวินโดว์ใหม่ ชื่อว่า sub ซึ่งเป็นส่วนหนึ่งของวินโดว์ orig

```
WINDOW          *win;
void            delwin(win);
```

ฟังก์ชันนี้จะทำการลบวินโดว์ win และปล่อยความจำ (free memory) ทั้งหมดที่เกี่ยวข้องกับวินโดว์ win

```
WINDOW          *win;
chtype          vert,hor;
void            box(win,vert,hor)
```

ฟังก์ชันนี้จะทำการวาดกรอบรอบวินโดว์ win โดยใช้ตัวอักษร vert สำหรับวาดกรอบตามแนวดิ่ง และ hor สำหรับวาดกรอบตามแนวราบ

ถ้า vert หรือ hor เป็น 0 box() จะใช้ตัวอักษร '¦' สำหรับวาดกรอบตามแนวดิ่ง และ '-' สำหรับวาดกรอบตามแนวราบ

```
WINDOW          *win;
int             wrefresh(win);
```

ฟังก์ชันนี้จะทำการปรับวินโดว์ win บนจอภาพให้ตรงกับวินโดว์ curscr

```
WINDOW          *win;
int             wstandout(win);
```

ฟังก์ชันนี้จะกำหนดคุณลักษณะทางจอภาพ (video attribute) ให้อยู่ในรูปโดดเด่น (standout) ของวินโดว์ win

```
WINDOW          *win;
int             wstandend(win);
```

ฟังก์ชันนี้จะทำการปรับคุณลักษณะทางจอภาพของวินโดว์ win ให้อยู่ในสภาพปกติ

```
WINDOW          *win;
int             wclear(win);
```

ฟังก์ชันนี้จะทำให้ข้อมูลที่อยู่ในวินโดว์ win ว่างเปล่า

2.6 ฟังก์ชันที่เกี่ยวข้องกับการอ่านและแสดงผลข้อมูล

```
WINDOW          *win;
chtype          ch;
int             waddch(win,ch);
```

ฟังก์ชันนี้จะนำตัวอักษร ch เข้าวินโดว์ win ที่ตำแหน่ง y,x ขณะนั้น

```
WINDOW          *win;
int             x,y;
char            *str;
int             mvwaddstr(win,y,x,str)
```

ฟังก์ชันนี้จะนำสตริง str เข้าไว้ที่ตำแหน่ง y,x ของวินโดว์ win

```
WINDOW          *win;
int             wgetch(win);
```

ฟังก์ชันนี้จะอ่านตัวอักษรจากวินโดว์ win บนเทอร์มินอลที่แสดงผลอยู่

```
WINDOW          *win;
int             x,y;
char            *fmt;
int             mvwprintw(win,y,x,fmt,[,arg]..);
```

ฟังก์ชันนี้จะแสดงสตริง fmt ตามรูปแบบที่กำหนดไว้ใน arg แสดงในวินโดว์ win ที่ตำแหน่ง y,x

2.7 การแปลชุดคำสั่ง (compile) โปรแกรมเคิร์ส

เนื่องจากว่าโปรแกรมเคิร์สเป็นโปรแกรมภาษา C ดังนั้นเราสามารถแปลชุดคำสั่งโปรแกรมเคิร์ส เหมือนโปรแกรมภาษา C ทั่วไปบนยูนิกซ์ แต่มีข้อเพิ่มเติมคือ ต้องใส่แฟ้ม /usr/lib/libcurses.a หรือ /lib/libcurses.a เข้าไปแล้วแต่ประเภทของยูนิกซ์ด้วย

ตัวอย่างเช่น ในการแปลชุดคำสั่งโปรแกรม foobar.c จะต้องใช้คำสั่งดังนี้

```
cc -o foobar foobar.c -lcurses
```

โดย cc เป็นโปรแกรมแปลชุดคำสั่งภาษา C (C Compiler) บนยูนิกซ์ และ foobar เป็นโปรแกรมกระทำการที่ได้ (execute program)