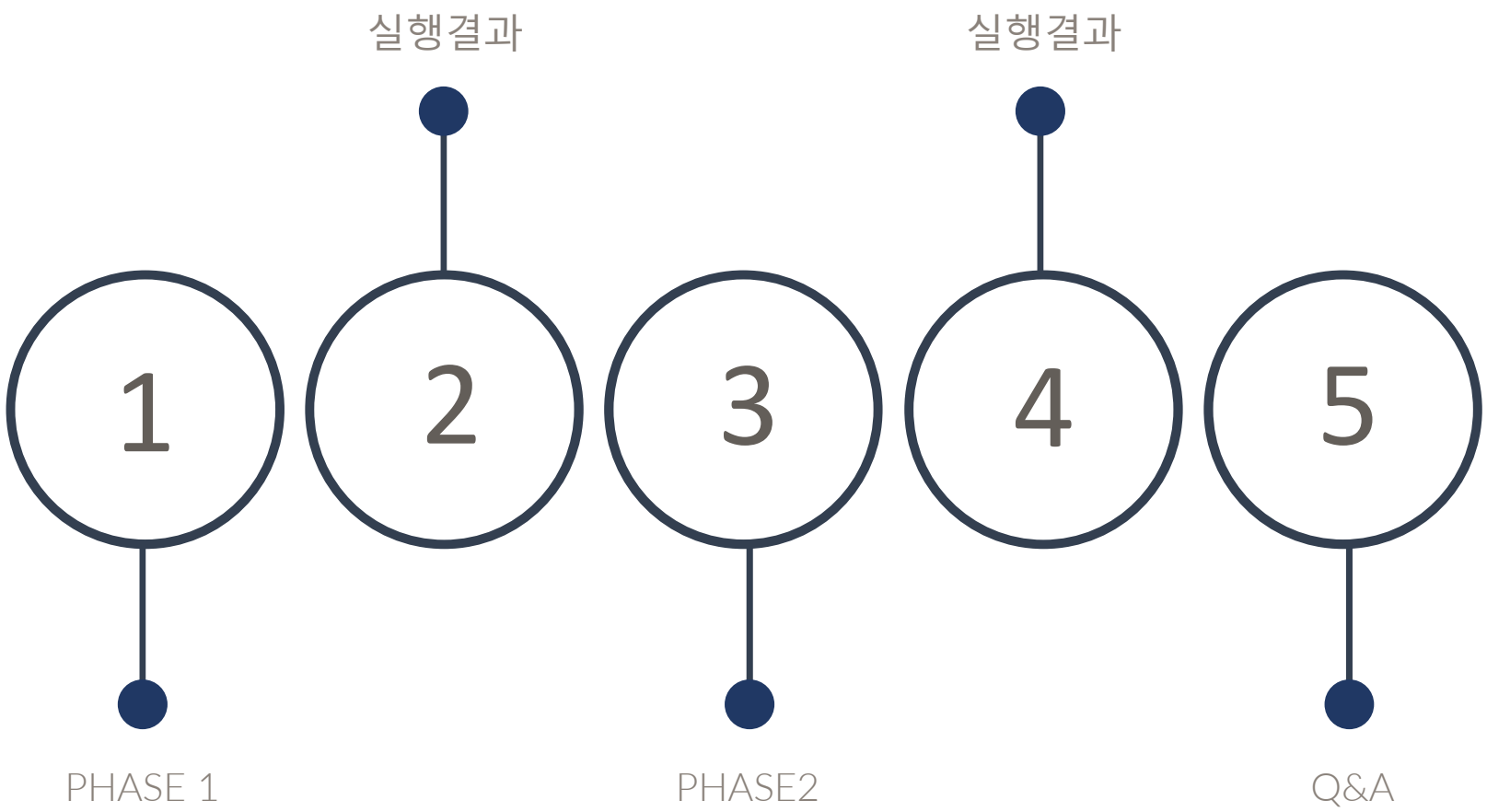
TITLE

EMBEDDED SYSTEM

DEVICE DRIVER

PROJECT STEPS



PHASE 1

소스 코드

소스 코드

변수 선언 단계

```c
#include <stdio.h>

int X = 80;
int Y = 24;
char fb[24][80]={
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ','*','*',' ',' ',' ','*','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*','*','*','*','*','*','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ','*',' ',' ','*',' ','*',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ','*',' ',' ',' ',' ','*',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','*','*','*','*','*','*','*',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},
{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '}};
int cursor[2] = {0,0};
int i=0; int j=0; int k=0; int m=0;
```

소스 코드

함수 선언 단계(1/3)

INITIALIZE

- 모든 변수들을 0으로 초기화
- ERASE SCREEN, CURSOR HOME

```
void initialize(){
    i=0; j=0; k=0; m=0; cursor[0]=0; cursor[1]=0;
    putchar('\033'); putchar('['); putchar('2'); putchar('J');
    putchar('\033'); putchar('['); putchar(';'); putchar('H');
}
```

FORWARD

*cursor[0] = 열 , cursor[1] = 행

- count만큼 커서를 오른쪽으로 이동

```
void forward(int count){
    for(k=0 ; k<count ; k++){
        cursor[0]++;
        if(cursor[0] >= X){
            cursor[0] = X-1;
            break;
        } else {
            putchar('\x1B'); putchar('['); putchar('1'); putchar('C');
        }
    }
}
```

소스 코드

함수 선언 단계(2/3)

BACKWARD

*cursor[0] = 열 , cursor[1] = 행

- count만큼 커서를 왼쪽으로 이동

```
void backward(int count){
    for(k=0 ; k<count ; k++){
        cursor[0]--;
        if(cursor[0] < 0){
            cursor[0] = 0;
            break;
        } else {
            putchar('\x1B'); putchar('['); putchar('1'); putchar('D');
        }
    }
}
```

UP

*cursor[0] = 열 , cursor[1] = 행

- count만큼 커서를 위로 이동

```
void up(int count){
    for(k=0 ; k<count ; k++){
        cursor[1]--;
        if(cursor[1] < 0){
            cursor[1] = 0;
            break;
        } else {
            putchar('\x1B'); putchar('['); putchar('1'); putchar('A');
        }
    }
}
```

소스 코드

함수 선언 단계(3/3)

DOWN

*cursor[0] = 열 , cursor[1] = 행

- count만큼 커서를 아래로 이동

```
void down(int count){
    for(k=0 ; k<count ; k++){
        cursor[1]++;
        if(cursor[1] >= Y){
            cursor[1] = Y-1;
            break;
        } else {
            putchar('\x1B'); putchar('['); putchar('1'); putchar('B');
        }
    }
}
```

Cset

*cursor[0] = 열 , cursor[1] = 행

- pos_x, pos_y를 통해 해당 위치로 커서 이동

```
void Cset(int pos_x, int pos_y){
    cursor[1] = 0; cursor[0] = 0;
    putchar('\033'); putchar('['); putchar('0'); putchar(';'); putchar('0'); putchar('f');
    forward(pos_x);
    down(pos_y);

}
```

소스 코드

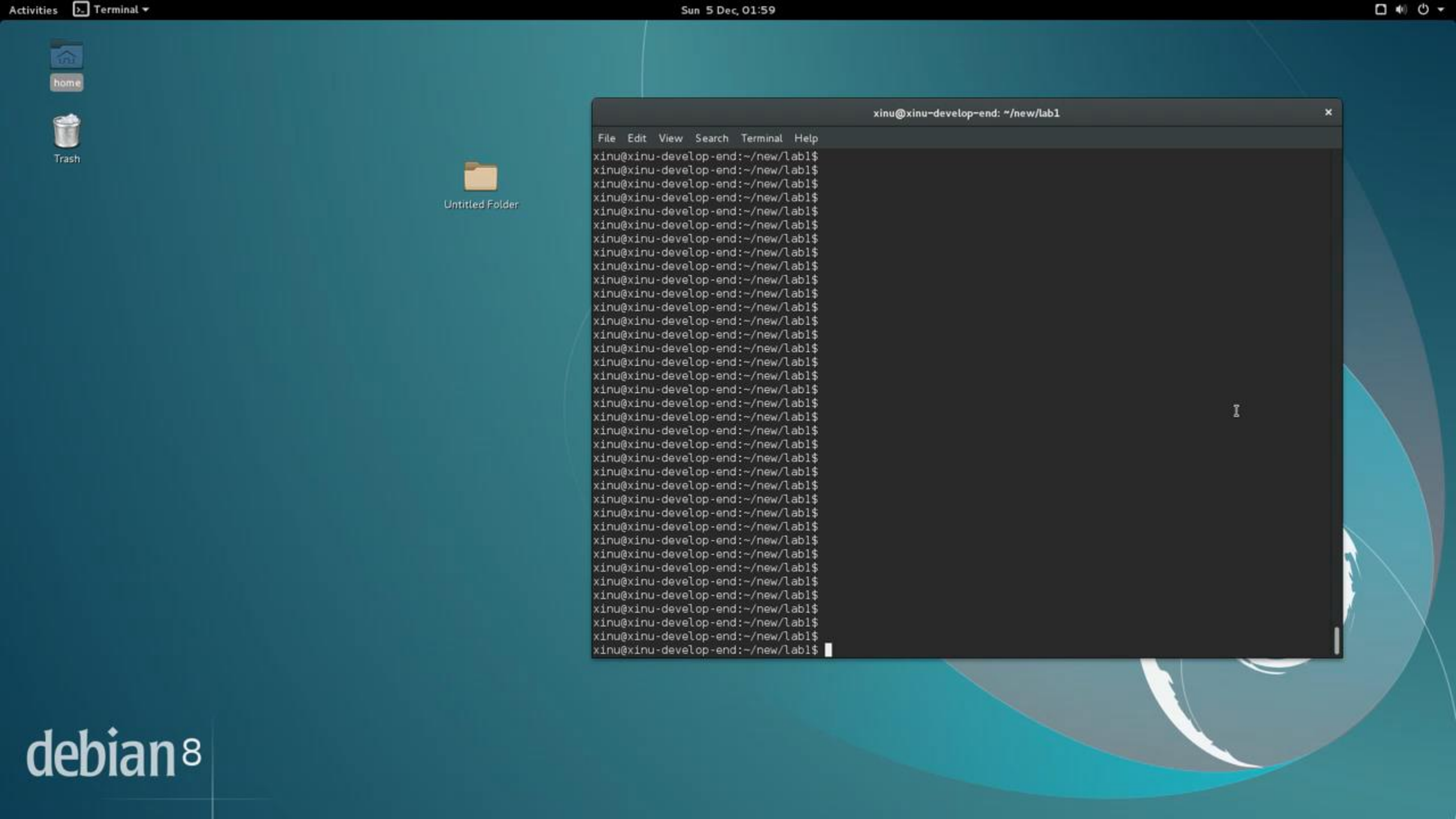
MAIN

```
void main(void)
{
    initialize();
    for(i=0; i<X ; i++){
        for(j=0; j<Y ; j++){
            Cset(j,i);
            printf("%s",&fb[cursor[1]][cursor[0]]);
        }
        printf("\n");
    }
}
```

MAIN

1. INITIALIZE 함수를 통해서 변수 초기화를 진행
2. Cset을 통해, 커서 위치를 한칸씩 이동
3. 해당 커서 위치의 문자출력

실행 결과


Activities
Terminal
Sun 5 Dec, 01:59
home
Trash
Untitled Folder
xinu@xinu-develop-end: ~/new/lab1
File Edit View Search Terminal Help
xinu@xinu-develop-end:~/new/lab1$
debian 8

PHASE 2

소스 코드

PHASE 2

FBINIT.C

FBINIT

*cursor_row = 행　　cursor_col = 열
fb_x = framebuffer의 행의 위치　　fb_y = 열의 위치

- 각 변수들을 선언과 동시에 초기화

```c
#include <xinu.h>

#include <stdio.h>
#include <string.h>

#define N 16

char framebuffer[16][16];  /* shared buffer */
int fb_mutex;
int cursor_row,cursor_col;
int fb_x,fb_y;

devcall fbinit (
      struct dentry *devptr     /* Entry in device switch table */
    )
{
    fb_mutex = semcreate(N);
    cursor_row = cursor_col = 0;
    fb_x = fb_y = 0;

    return OK;
}
```

PHASE 2

FBOPEN.C

FBOPEN

- ERASE SCREEN, CURSOR HOME

```
#include <xinu.h>

devcall fbopen (
      struct dentry *devptr     /* Entry in device switch table */
    )
{
    putchar('\033');putchar('[');putchar('2');putchar('J'); // erase screen
    putchar('\033');putchar('[');putchar(';');putchar('H'); //cursor home
    return OK;
}
```

PHASE 2

FBCTL.C(1/4)

FBCTL

*COMMAND = 원하는 command
MOVE_X = 움직일 x의 양
MOVE_Y = 움직일 y의 양

- 변수 선언

```
#include <xinu.h>
#include <string.h>
extern int fb_x,fb_y;
extern int cursor_row,cursor_col; //current cursor location

devcall fbctl (
    struct dentry   *devptr,
    char* command, //ctl command
    int32 move_x, //x move count
    int32 move_y //y move count
    )
{
    cursor_row = 0; cursor_col=0;
    int i = 0; int j = 0;
```

PHASE 2

FBCTL.C(2/4)

FBCTL

*cursor_COL = 열 , cursor_ROW = 행

- PHASE1과 동일
- 차이점은 원하는 키워드가 들어오면

해당 명령어를 실행시켜주는 역할

```
if(!strncmp(command,"forward", 8)){
    cursor_col += 1;
    putchar('\x1B'); putchar('['); putchar('1'); putchar('C');
} else if(!strncmp(command,"backward", 9)){
    cursor_col -= 1;
    putchar('\x1B'); putchar('['); putchar('1'); putchar('D');
} else if(!strncmp(command,"up", 3)){
    cursor_row -= 1;
    putchar('\x1B'); putchar('['); putchar('1'); putchar('A');
} else if(!strncmp(command,"down", 5)){
    cursor_row += 1;
    putchar('\x1B'); putchar('['); putchar('1'); putchar('B');
} else if(!strncmp(command,"next_line", 10)){
    cursor_col = 0;
    cursor_row += 1;
    for(i=0 ; i<cursor_row ; i++){
    putchar('\x1B'); putchar('['); putchar('1'); putchar('D'); // line_down
    putchar('\x1B'); putchar('['); putchar('1'); putchar('B');
    }
```

PHASE 2

FBCTL.C(3/4)

FBCTL

*cursor_COL = 열 , cursor_ROW = 행

- move 커맨드를 통해, 원하는 위치로 커서 이동

```
    } else if(!strncmp(command,"move", 5)){
        cursor_col = 0; cursor_row = 0;
        putchar('\033'); putchar('['); putchar(';'); putchar('H');
        for(i=0 ; i < move_x ; i++){
            cursor_col++;
            if(cursor_col >= 16){
                cursor_col = 15;
                break;
            }
            else {
                putchar('\x1B'); putchar('['); putchar('1'); putchar('C');
            }
        }

        for(j=0 ; j < move_y ; j++){
            cursor_row++;
            if(cursor_row >= 16){
                cursor_row = 15;
                break;
            }
            else {
                putchar('\x1B'); putchar('['); putchar('1'); putchar('C');
            }
        }
    return OK;
    }
}
```

PHASE 2

FBCTL.C(4/4)

FBCTL

*cursor_COL = 열 , cursor_ROW = 행

- reset 커맨드를 통해, 커서값과 커서를 HOME으로 이동

```
    } else if(!strncmp(command, "reset", 6)) {
        cursor_col = 0; cursor_row = 0;
        putchar('\033'); putchar('['); putchar(';'); putchar('H');

    }
}
```

PHASE 2

FBPUTC.C

FBPUTC

- Framebuffer에 매개변수로 입력 받은 개별 문자를 삽입

```
#include <xinu.h>

extern char framebuffer[16][16];  /* shared buffer */
extern int cursor_col,cursor_row;
extern int fb_x,fb_y;

#define     N   16

devcall fbputc (
                struct dentry *devptr,
                char ch
                )
{

        framebuffer[fb_x][fb_y] = ch;
    fb_y++;
        if (fb_y>= N){
            fb_y = 0;
            fb_x++;
            if(fb_x>=N){
                return SYSERR;
            }
        }

    return OK;
}
```

PHASE 2

FBWRITE.C

FBWRITE

- 배열로 된 입력 값을 framebuffer에 입력

```c
#include <xinu.h>
extern char framebuffer[16][16];  /* shared buffer */
extern int cursor_col,cursor_row;
extern int fb_x,fb_y;

devcall fbwrite(
                struct dentry *devptr,
                char *buff[16]
                )
{
    int i,j;

    for(i=0;i<16;i++)
    {
        char ch[16]={};
        for(j = 0;j < 16; j++)
        {
            ch[j] = buff[i][j];
        }
        for(j = 0;j < 16; j++)
        {
            fbputc(devptr,ch[j]);
        }

    }
    return OK;
}
```

PHASE 2

FBFLUSH.C

FBFLUSH

- Device의 framebuffer의 문자들을 tty화면에 출력

```
#include <xinu.h>
extern char framebuffer[16][16];  /* shared buffer */

int cursor_row,cursor_col;
int fb_x,fb_y;


devcall fbflush (
     struct dentry *devptr /* Entry in device switch table */
     /*char    (*fb)[16]*/         /* Buffer to hold bytes     */
          /* Max bytes to read        */
   )
{
   int i, j;
   sleep(2);
   for(i = 0; i < 16; i++)
   {
      if (i == 0){
         putchar('\033'); putchar('['); putchar('2'); putchar('J'); //erase screen
         putchar('\033'); putchar('['); putchar(';'); putchar('H'); //cursor homing
      }
      for(j = 0; j < 16; j++)
      {
         fbctl(CONSOLE, "reset",0,0); //cursor reset
         fbctl(CONSOLE,"move",j,i); //cursor move
         putc(CONSOLE, framebuffer[cursor_row][cursor_col]); // putc in current cursor
         sleep(0.3);
      }

   }
   putc(CONSOLE, '\n');
   return OK;
}
```

PHASE 2

XSH_PS.C(1/3)

XSH_PS_prod(1)

- 입력 받을 배열 16*16 배열 준비
- Device init, open
- 배열을 putc로 buffer에 입력
- Flush로 출력

```
void prod2()
{

    int i,j;
    char c;

    char *fb[16] = {

            {"     *******      "},
            {"        *         "},
            {"      *****       "},
            {"     **  *  **     "},
            {"     *   *   *     "},
            {"     *   *   *     "},
            {"     *   *   *     "},
            {"     *   *   *     "},
            {"      *****        "},
            {"        *          "},
            {"        *          "},
            {"        *          "},
            {"        *          "},
            {"        *          "},
            {"        *          "},
            {"        *          "}


    };

        init(FBUF15);
        fbopen(FBUF15);
        for(i = 0; i < 16; i++)
        {
                for (j = 0; j < 16; j++)
                {
                        putc(FBUF15, fb[i][j]);
                }
        }
        fbflush(FBUF15);

}
```

PHASE 2

XSH_PS.C(2/3)

XSH_PS_prod(2)

- 같은 작업에 putc대신 write 사용

```
void prod2()
{

    int i,j;
    char c;

    char *fb[16] = {

        {"    *******     "},
        {"       *        "},
        {"     *****      "},
        {"    ** * **     "},
        {"    *  *  *     "},
        {"    *  *  *     "},
        {"    *  *  *     "},
        {"    *  *  *     "},
        {"     *****      "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "}


    };
    init(FBUF15);
    fbopen(FBUF15);
    fbwrite(FBUF15,fb);
    fbflush(FBUF15);

}
```

```
void prod3()
{

    int i,j;
    char c;

    char *fb[16] = {

        {"       *        "},
        {"     *   *      "},
        {"   *       *    "},
        {" *           *  "},
        {"*             * "},
        {"*             * "},
        {"*             * "},
        {"  ***********    "},
        {"    *   *   *    "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "},
        {"       *        "}


    };
    init(FBUF15);
    fbopen(FBUF15);
    fbwrite(FBUF15,fb);
    fbflush(FBUF15);

}
```

PHASE 2

XSH_PS.C(3/3)

XSH_PS_ring

- prod2, prod3를 create하는 ring함수를 정의
- Ring을 실행할 호출명 지정

```
} else if (nargs == 2 && strncmp(args[1], "ES2", 4) == 0) {
    ring(1);
    return 0;
} else if (nargs == 2 && strncmp(args[1], "ES3", 4) == 0) {
    ring(2);
    return 0;
}
```

```
void ring(int num)
{
    switch(num)
    {
    case 1:
        resume( create(prod2, 1000, 20, "prod", 0, NULL));
        break;
    case 2:
        resume( create(prod3, 1000, 20, "prod", 0, NULL));
        break;
    }

}
```

PHASE 2

PROTOTYPES.H, MAKEFILE, CONF

PROTOTYPES

- 각 함수들을 헤더에 extern 선언

MAKEFILE

- rebuild를 위해 설정 추가

CONFIGURATION

- rebuild를 위해 설정 추가

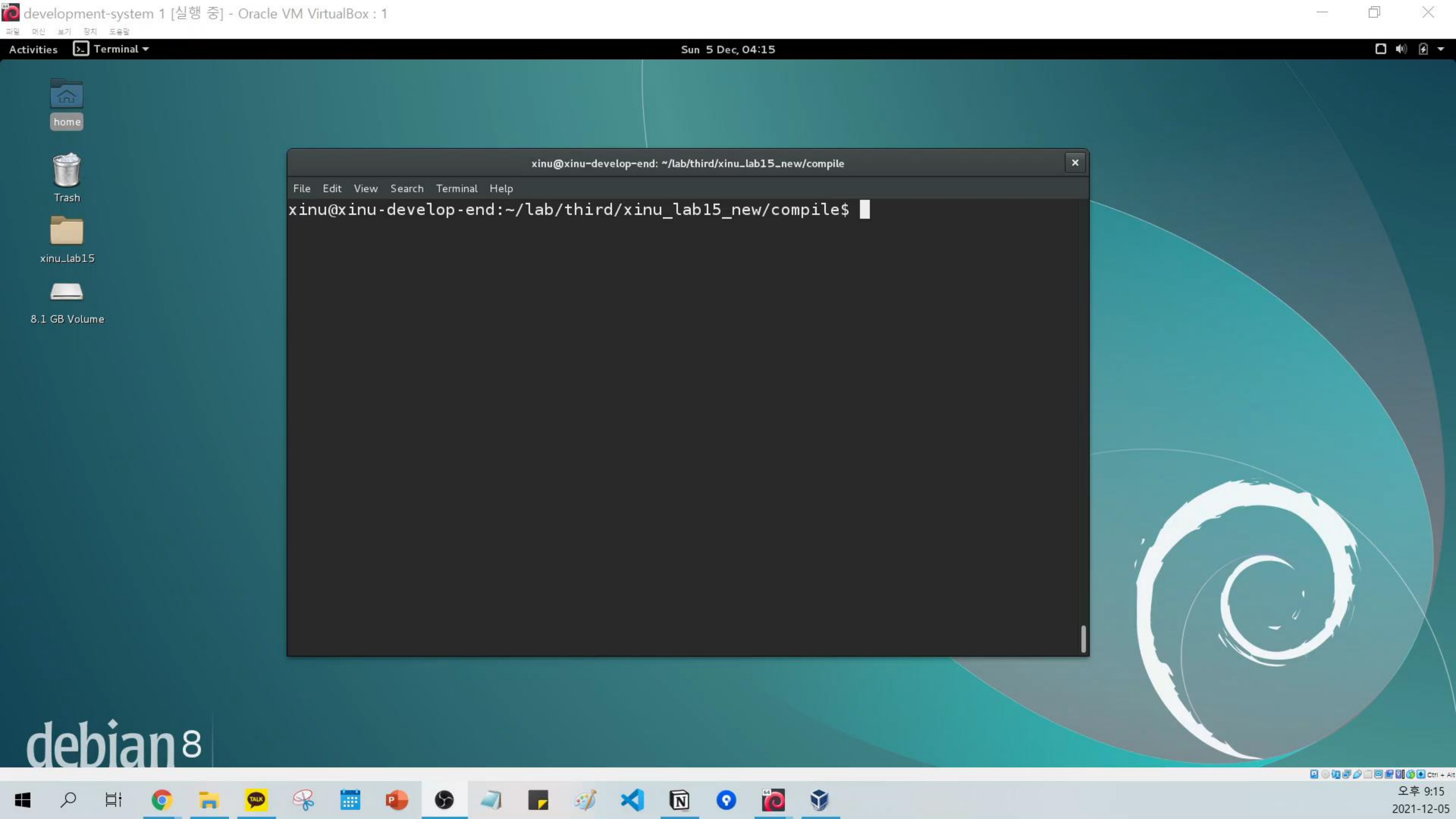
```
extern  devcall fbctl(struct dentry *, char *, int32, int32);
extern  devcall fbopen(struct dentry *);
extern  devcall fbinit(struct dentry *);
extern  devcall fbputc(struct dentry *, char);
extern  devcall fbflush(struct dentry *);
extern  devcall fbwrite(struct dentry *, char *, int32);
extern  devcall fbgetc(struct dentry *);
```

```
REBUILDFLAGS    =   -s $(TOPDIR)/system  debug.c    \
            -s $(TOPDIR)/lib            \
            -s $(TOPDIR)/device/tty         \
            -s $(TOPDIR)/device/nam         \
            -s $(TOPDIR)/device/eth         \
            -s $(TOPDIR)/device/rds         \
            -s $(TOPDIR)/device/ram         \
            -s $(TOPDIR)/device/lfs         \
            -s $(TOPDIR)/device/rfs         \
            -s $(TOPDIR)/device/fbf         \
            -s $(TOPDIR)/net  'arp_dump*'       \
               'dhcp_dump*'  pxe.c      \
            -s $(TOPDIR)/shell  'xsh_rdstest*'
```

```
/* type of a local file pseudo-device */
fbf:
    on nothing
        -i fbinit   -o ioerr    -c fbopen
        -r fbflush  -p fbputc   -g fbgetc
        -w fbwrite  -s ioerr    -n fbctl
        -intr ionull

/* type of namespace device */
```

실행 결과


development-system 1 [실행 중] - Oracle VM VirtualBox : 1
파일 머신 보기 장치 도움말
Activities
Terminal
Sun 5 Dec, 04:15
home
Trash
xinu_lab15
8.1 GB Volume
xinu@xinu-develop-end: ~/lab/third/xinu_lab15_new/compile
File Edit View Search Terminal Help
xinu@xinu-develop-end:~/lab/third/xinu_lab15_new/compile$
debian 8
오후 9:15
2021-12-05

THANK YOU