|  |
| --- |
| // 本代码仅供研究[luther.gliethttp]  #include <stdio.h> #include <sys/types.h> #include <sys/ipc.h> #include <string.h>  #define QT\_VFB\_MOUSE\_PIPE "/tmp/.qtvfb\_mouse-%d" #define QT\_VFB\_KEYBOARD\_PIPE "/tmp/.qtvfb\_keyboard-%d" typedef struct \_RECT {     int left;int top;int right;int bottom; } RECT;  struct QVFbHeader {     int width;     int height;     int depth;     int linestep;     int dataoffset;     RECT update;     unsigned char dirty;     int numcols;     unsigned int clut[256]; };  struct \_fb {     struct QVFbHeader \*hdr;     unsigned char \*data;     int size;     int BitsPerPixel;     int BytesPerPixel;     int linesize;     int Rmask;     int Gmask;     int Bmask;     int Amask; };  unsigned char\* shmrgn;  void lu\_printf\_hdr(struct QVFbHeader \*hdr, const char \*prefix) {     printf("=====[luther.gliethttp] %s=====\n"            "width\t= %d\n"            "height\t= %d\n"            "depth\t= %d\n"            "linestep= %d\n"            "dataoffs= %d\n"            "dirty\t= %d\n"            "numcols\t= %d\n"            "left\t= %d\n"            "top\t= %d\n"            "right\t= %d\n"            "bottom\t= %d\n",            prefix,            hdr->width,            hdr->height,            hdr->depth,            hdr->linestep,            hdr->dataoffset,            hdr->dirty,            hdr->numcols,            hdr->update.left,            hdr->update.top,            hdr->update.right,            hdr->update.bottom); }  int main(void) {     char file [50];     int display;     key\_t key;     int shmid;     struct QVFbHeader \*hdr;     struct \_fb fb;     int i;     RECT rect;          display = 0;      sprintf (file, QT\_VFB\_MOUSE\_PIPE, display);     key = ftok (file, 'b');      shmid = shmget (key, 0, 0);     if (shmid != -1)         shmrgn = (unsigned char \*)shmat (shmid, 0, 0);     else {         printf("[luther.gliethttp] shmid=-1\n");         return -1;     }          fb.hdr = hdr = (struct QVFbHeader \*) shmrgn;      lu\_printf\_hdr(fb.hdr,"1111111111");      fb.data = (char \*)hdr + hdr->dataoffset;     fb.linesize = hdr->linestep;     fb.BitsPerPixel = hdr->depth;     switch (fb.BitsPerPixel) {         case 8:             fb.BytesPerPixel = 1;             hdr->numcols = 256;             break;         case 12:             fb.BitsPerPixel = 16;             fb.BytesPerPixel = 2;             fb.Rmask = 0x00000F00;             fb.Gmask = 0x000000F0;             fb.Bmask = 0x0000000F;             break;         case 16:             fb.BytesPerPixel = 2;             fb.Rmask = 0x0000F800; // R:G:B=5:6:5模式             fb.Gmask = 0x000007E0;             fb.Bmask = 0x0000001F;             break;         case 32:             fb.BytesPerPixel = 4;             fb.Rmask = 0x00FF0000;             fb.Gmask = 0x0000FF00;             fb.Bmask = 0x000000FF;             break;         default:             printf("[luther.gliethttp] bits error!\n");             return -1;     }          for (i = 0; i < fb.hdr->height; i++) {         memset(fb.data + i\*fb.linesize, 0x9a, fb.linesize);     }      for (i = 1; i < (6\*fb.linesize+1); i++) {         printf("%04x ", ((unsigned short \*)fb.data)[i]);         if ((i & 0xf) == 0)             printf("\n");         // memset(&fb.data[i\*fb.linesize], 0x23, fb.linesize);     }     printf("\n");     rect.left = 0;     rect.top = 0;     rect.right = fb.hdr->width/2; // 仅刷新一半,如果拖动qvfb窗体,另外一个可以出现[luther.gliethttp]     rect.bottom = fb.hdr->height;      hdr->update = rect;     hdr->dirty = 1;     lu\_printf\_hdr(fb.hdr,"22222222");     sleep(5);     lu\_printf\_hdr(fb.hdr,"33333333");          rect.right = fb.hdr->width; // 让整个屏幕所在矩形都执行刷新          for (;;) {          for (i = 0; i < fb.hdr->height; i += 120) {             memset(fb.data + i\*fb.linesize, rand() & 0xff, fb.linesize\*120);         }          hdr->update = rect;         hdr->dirty = 1; // qvfb软件读取到该共享内存中dirty标志置位后刷新update所描述的矩形空间[luther.gliethttp]         usleep(300\*1000);     }     shmdt (fb.hdr);     return 0; } |

|  |
| --- |
| // 本代码仅供个人学习研究,版权归MiniGUI所有.[luther.gliethttp] #include <stdio.h> #include <fcntl.h> #include <sys/select.h> #define QT\_VFB\_MOUSE\_PIPE "/tmp/.qtvfb\_mouse-%d" // pipe管道文件[luther.gliethttp] #define QT\_VFB\_KEYBOARD\_PIPE "/tmp/.qtvfb\_keyboard-%d" // pipe管道文件[luther.gliethttp]  #define SCANCODE\_ESCAPE 1  #define SCANCODE\_1 2 #define SCANCODE\_2 3 #define SCANCODE\_3 4 #define SCANCODE\_4 5 #define SCANCODE\_5 6 #define SCANCODE\_6 7 #define SCANCODE\_7 8 #define SCANCODE\_8 9 #define SCANCODE\_9 10 #define SCANCODE\_0 11  #define SCANCODE\_MINUS 12 #define SCANCODE\_EQUAL 13  #define SCANCODE\_BACKSPACE 14 #define SCANCODE\_TAB 15  #define SCANCODE\_Q 16 #define SCANCODE\_W 17 #define SCANCODE\_E 18 #define SCANCODE\_R 19 #define SCANCODE\_T 20 #define SCANCODE\_Y 21 #define SCANCODE\_U 22 #define SCANCODE\_I 23 #define SCANCODE\_O 24 #define SCANCODE\_P 25 #define SCANCODE\_BRACKET\_LEFT 26 #define SCANCODE\_BRACKET\_RIGHT 27  #define SCANCODE\_ENTER 28  #define SCANCODE\_LEFTCONTROL 29  #define SCANCODE\_A 30 #define SCANCODE\_S 31 #define SCANCODE\_D 32 #define SCANCODE\_F 33 #define SCANCODE\_G 34 #define SCANCODE\_H 35 #define SCANCODE\_J 36 #define SCANCODE\_K 37 #define SCANCODE\_L 38 #define SCANCODE\_SEMICOLON 39 #define SCANCODE\_APOSTROPHE 40 #define SCANCODE\_GRAVE 41  #define SCANCODE\_LEFTSHIFT 42 #define SCANCODE\_BACKSLASH 43  #define SCANCODE\_Z 44 #define SCANCODE\_X 45 #define SCANCODE\_C 46 #define SCANCODE\_V 47 #define SCANCODE\_B 48 #define SCANCODE\_N 49 #define SCANCODE\_M 50 #define SCANCODE\_COMMA 51 #define SCANCODE\_PERIOD 52 #define SCANCODE\_SLASH 53  #define SCANCODE\_RIGHTSHIFT 54 #define SCANCODE\_KEYPADMULTIPLY 55  #define SCANCODE\_LEFTALT 56 #define SCANCODE\_SPACE 57 #define SCANCODE\_CAPSLOCK 58  #define SCANCODE\_F1 59 #define SCANCODE\_F2 60 #define SCANCODE\_F3 61 #define SCANCODE\_F4 62 #define SCANCODE\_F5 63 #define SCANCODE\_F6 64 #define SCANCODE\_F7 65 #define SCANCODE\_F8 66 #define SCANCODE\_F9 67 #define SCANCODE\_F10 68  #define SCANCODE\_NUMLOCK 69 #define SCANCODE\_SCROLLLOCK 70  #define SCANCODE\_KEYPAD7 71 #define SCANCODE\_CURSORUPLEFT 71 #define SCANCODE\_KEYPAD8 72 #define SCANCODE\_CURSORUP 72 #define SCANCODE\_KEYPAD9 73 #define SCANCODE\_CURSORUPRIGHT 73 #define SCANCODE\_KEYPADMINUS 74 #define SCANCODE\_KEYPAD4 75 #define SCANCODE\_CURSORLEFT 75 #define SCANCODE\_KEYPAD5 76 #define SCANCODE\_KEYPAD6 77 #define SCANCODE\_CURSORRIGHT 77 #define SCANCODE\_KEYPADPLUS 78 #define SCANCODE\_KEYPAD1 79 #define SCANCODE\_CURSORDOWNLEFT 79 #define SCANCODE\_KEYPAD2 80 #define SCANCODE\_CURSORDOWN 80 #define SCANCODE\_KEYPAD3 81 #define SCANCODE\_CURSORDOWNRIGHT 81 #define SCANCODE\_KEYPAD0 82 #define SCANCODE\_KEYPADPERIOD 83  #define SCANCODE\_LESS 86  #define SCANCODE\_F11 87 #define SCANCODE\_F12 88  #define SCANCODE\_KEYPADENTER 96 #define SCANCODE\_RIGHTCONTROL 97 #define SCANCODE\_CONTROL 97 #define SCANCODE\_KEYPADDIVIDE 98 #define SCANCODE\_PRINTSCREEN 99 #define SCANCODE\_RIGHTALT 100 #define SCANCODE\_BREAK 101 /\* Beware: is 119 \*/ #define SCANCODE\_BREAK\_ALTERNATIVE 119 /\* on some keyboards! \*/  #define SCANCODE\_HOME 102 #define SCANCODE\_CURSORBLOCKUP 103 /\* Cursor key block \*/ #define SCANCODE\_PAGEUP 104 #define SCANCODE\_CURSORBLOCKLEFT 105 /\* Cursor key block \*/ #define SCANCODE\_CURSORBLOCKRIGHT 106 /\* Cursor key block \*/ #define SCANCODE\_END 107 #define SCANCODE\_CURSORBLOCKDOWN 108 /\* Cursor key block \*/ #define SCANCODE\_PAGEDOWN 109 #define SCANCODE\_INSERT 110 #define SCANCODE\_REMOVE 111  #define SCANCODE\_PAUSE 119  #define SCANCODE\_POWER 120 #define SCANCODE\_SLEEP 121 #define SCANCODE\_WAKEUP 122  #define SCANCODE\_LEFTWIN 125 #define SCANCODE\_RIGHTWIN 126 #define SCANCODE\_MENU 127  #define SCANCODE\_LEFTBUTTON 0x1000 #define SCANCODE\_RIGHTBUTTON 0x2000 #define SCANCODE\_MIDDLBUTTON 0x4000  #define NOBUTTON 0x0000 #define LEFTBUTTON 0x0001 #define RIGHTBUTTON 0x0002 #define MIDBUTTON 0x0004 #define MOUSEBUTTONMASK 0x00FF    #define SHIFTBUTTON 0x0100 #define CONTROLBUTTON 0x0200 #define ALTBUTTON 0x0400 #define METABUTTON 0x0800 #define KEYBUTTONMASK 0x0FFF #define KEYPAD 0x4000  struct QVFbKeyData {     #define NR\_KEYS 128     unsigned int unicode;     unsigned int modifiers;     unsigned char press;     unsigned char repeat; };  typedef struct \_POINT {     int x;int y; } POINT, \*PPOINT;  static int mouse\_fd = -1; static int kbd\_fd = -1; static POINT mouse\_pt; static int mouse\_buttons; static struct QVFbKeyData kbd\_data; static unsigned char kbd\_state [NR\_KEYS]; static unsigned char keycode\_scancode [256];  // 本代码源自libminigui-1.6.10/src/ial/qvfb.c  static void init\_code\_map (void) {     keycode\_scancode [0x00] = SCANCODE\_ESCAPE;      keycode\_scancode [0x29] = SCANCODE\_0;     keycode\_scancode [0x21] = SCANCODE\_1;     keycode\_scancode [0x40] = SCANCODE\_2;     keycode\_scancode [0x23] = SCANCODE\_3;     keycode\_scancode [0x24] = SCANCODE\_4;     keycode\_scancode [0x25] = SCANCODE\_5;     keycode\_scancode [0x5E] = SCANCODE\_6;     keycode\_scancode [0x26] = SCANCODE\_7;     keycode\_scancode [0x2A] = SCANCODE\_8;     keycode\_scancode [0x28] = SCANCODE\_9;      keycode\_scancode [0x30] = SCANCODE\_0;     keycode\_scancode [0x31] = SCANCODE\_1;     keycode\_scancode [0x32] = SCANCODE\_2;     keycode\_scancode [0x33] = SCANCODE\_3;     keycode\_scancode [0x34] = SCANCODE\_4;     keycode\_scancode [0x35] = SCANCODE\_5;     keycode\_scancode [0x36] = SCANCODE\_6;     keycode\_scancode [0x37] = SCANCODE\_7;     keycode\_scancode [0x38] = SCANCODE\_8;     keycode\_scancode [0x39] = SCANCODE\_9;     keycode\_scancode [0x2D] = SCANCODE\_MINUS;     keycode\_scancode [0x5F] = SCANCODE\_MINUS;     keycode\_scancode [0x3D] = SCANCODE\_EQUAL;     keycode\_scancode [0x2B] = SCANCODE\_EQUAL;     keycode\_scancode [0x03] = SCANCODE\_BACKSPACE;     keycode\_scancode [0x01] = SCANCODE\_TAB;     keycode\_scancode [0x51] = SCANCODE\_Q;     keycode\_scancode [0x57] = SCANCODE\_W;     keycode\_scancode [0x45] = SCANCODE\_E;     keycode\_scancode [0x52] = SCANCODE\_R;     keycode\_scancode [0x54] = SCANCODE\_T;     keycode\_scancode [0x59] = SCANCODE\_Y;     keycode\_scancode [0x55] = SCANCODE\_U;     keycode\_scancode [0x49] = SCANCODE\_I;     keycode\_scancode [0x4F] = SCANCODE\_O;     keycode\_scancode [0x50] = SCANCODE\_P;     keycode\_scancode [0x5B] = SCANCODE\_BRACKET\_LEFT;     keycode\_scancode [0x7B] = SCANCODE\_BRACKET\_LEFT;     keycode\_scancode [0x5D] = SCANCODE\_BRACKET\_RIGHT;     keycode\_scancode [0x7D] = SCANCODE\_BRACKET\_RIGHT;     keycode\_scancode [0x04] = SCANCODE\_ENTER;     keycode\_scancode [0x41] = SCANCODE\_A;     keycode\_scancode [0x53] = SCANCODE\_S;     keycode\_scancode [0x44] = SCANCODE\_D;     keycode\_scancode [0x46] = SCANCODE\_F;     keycode\_scancode [0x47] = SCANCODE\_G;     keycode\_scancode [0x48] = SCANCODE\_H;     keycode\_scancode [0x4A] = SCANCODE\_J;     keycode\_scancode [0x4B] = SCANCODE\_K;     keycode\_scancode [0x4C] = SCANCODE\_L;      keycode\_scancode [0x3A] = SCANCODE\_SEMICOLON;     keycode\_scancode [0x3B] = SCANCODE\_SEMICOLON;      keycode\_scancode [0x27] = SCANCODE\_APOSTROPHE;     keycode\_scancode [0x22] = SCANCODE\_APOSTROPHE;     keycode\_scancode [0x60] = SCANCODE\_GRAVE;     keycode\_scancode [0x7E] = SCANCODE\_GRAVE;      keycode\_scancode [0x20] = SCANCODE\_SPACE;      keycode\_scancode [0x5C] = SCANCODE\_BACKSLASH;     keycode\_scancode [0x7C] = SCANCODE\_BACKSLASH;     keycode\_scancode [0x5A] = SCANCODE\_Z;     keycode\_scancode [0x58] = SCANCODE\_X;     keycode\_scancode [0x43] = SCANCODE\_C;     keycode\_scancode [0x56] = SCANCODE\_V;     keycode\_scancode [0x42] = SCANCODE\_B;     keycode\_scancode [0x4E] = SCANCODE\_N;     keycode\_scancode [0x4D] = SCANCODE\_M;     keycode\_scancode [0x2C] = SCANCODE\_COMMA;     keycode\_scancode [0x3C] = SCANCODE\_COMMA;     keycode\_scancode [0x2E] = SCANCODE\_PERIOD;     keycode\_scancode [0x3E] = SCANCODE\_PERIOD;     keycode\_scancode [0x2F] = SCANCODE\_SLASH;     keycode\_scancode [0x3F] = SCANCODE\_SLASH;      keycode\_scancode [0x09] = SCANCODE\_PRINTSCREEN;     keycode\_scancode [0x08] = SCANCODE\_BREAK;     keycode\_scancode [0x06] = SCANCODE\_INSERT;     keycode\_scancode [0x07] = SCANCODE\_REMOVE;     keycode\_scancode [0x10] = SCANCODE\_HOME;     keycode\_scancode [0x11] = SCANCODE\_END;     keycode\_scancode [0x16] = SCANCODE\_PAGEUP;     keycode\_scancode [0x17] = SCANCODE\_PAGEDOWN;     keycode\_scancode [0x13] = SCANCODE\_CURSORBLOCKUP;     keycode\_scancode [0x12] = SCANCODE\_CURSORBLOCKLEFT;     keycode\_scancode [0x14] = SCANCODE\_CURSORBLOCKRIGHT;     keycode\_scancode [0x15] = SCANCODE\_CURSORBLOCKDOWN; }  static unsigned char keycode\_to\_scancode (unsigned char keycode, char asscii) {     if(!asscii)     {         switch (keycode) {             case 0x30 ... 0x39:                 return SCANCODE\_F1 + keycode - 0x30;             case 0x3B:                 return SCANCODE\_F12;             case 0x20:                 return SCANCODE\_LEFTSHIFT;             case 0x21:                 return SCANCODE\_LEFTCONTROL;             case 0x23:                 return SCANCODE\_LEFTALT;             case 0x24:                 return SCANCODE\_CAPSLOCK;             case 0x25:                 return SCANCODE\_NUMLOCK;             case 0x26:                 return SCANCODE\_SCROLLLOCK;             case 0x3A:                 return SCANCODE\_F11;         }     }      return keycode\_scancode [keycode]; }  static int mouse\_update (void) {     int ret1, ret2;     POINT l\_mouse\_pt;     int l\_mouse\_buttons;      ret1 = read (mouse\_fd, &l\_mouse\_pt, sizeof (POINT));     ret2 = read (mouse\_fd, &l\_mouse\_buttons, sizeof (int));      if (ret1 == sizeof (POINT) && ret2 && sizeof (int)             && l\_mouse\_buttons < 0x08) {         mouse\_pt = l\_mouse\_pt;         mouse\_buttons = l\_mouse\_buttons;     }     else         return 0;      return 1; }  static void mouse\_getxy (int \*x, int\* y) {     \*x = mouse\_pt.x;     \*y = mouse\_pt.y; }  static int mouse\_getbutton (void) {     int buttons = 0;      if (mouse\_buttons & LEFTBUTTON)         buttons |= 1;     if (mouse\_buttons & RIGHTBUTTON)         buttons |= 4;     if (mouse\_buttons & MIDBUTTON)         buttons |= 2;      return buttons; }  static unsigned char nr\_changed\_keys = 0;  static int keyboard\_update (void) {     return nr\_changed\_keys; }  static int read\_key (void) {     static unsigned char last;     struct QVFbKeyData l\_kbd\_data;     int ret;     unsigned char scancode;      ret = read (kbd\_fd, &l\_kbd\_data, sizeof (struct QVFbKeyData));      if (ret == sizeof (struct QVFbKeyData)) {         kbd\_data = l\_kbd\_data;     }     else         return 0;      if (kbd\_data.repeat) {         return 0;     }      if (kbd\_data.unicode == 0 && !kbd\_data.press) {         kbd\_state [last] = 0;     }     else {         scancode = keycode\_to\_scancode ((kbd\_data.unicode >> 16) & 0x00FF,                      kbd\_data.unicode & 0x00FF);         kbd\_state [scancode] = kbd\_data.press ? 1 : 0;         last = scancode;     }      nr\_changed\_keys = last + 1;     return 1; } static const char\* keyboard\_getstate (void) {     return (char\*)kbd\_state; }   int main(void) {     char file [50];     int display = 0;     fd\_set rfds;      /\* open mouse pipe \*/     sprintf (file, QT\_VFB\_MOUSE\_PIPE, display);     if ((mouse\_fd = open (file, O\_RDONLY)) < 0) {         fprintf (stderr, "QVFB IAL engine: can not open mouse pipe.\n");         return -1;     }     /\* open keyboard pipe \*/     sprintf (file, QT\_VFB\_KEYBOARD\_PIPE, display);     if ((kbd\_fd = open (file, O\_RDONLY)) < 0) {         fprintf (stderr, "QVFB IAL engine: can not open keyboard pipe.\n");         return -1;     }          init\_code\_map();          for (;;) {         FD\_ZERO(&rfds);         FD\_SET(mouse\_fd, &rfds);         FD\_SET(kbd\_fd, &rfds);                  if ( select(FD\_SETSIZE, &rfds, 0, 0, 0) > 0 ) {             if ( mouse\_fd >= 0 && FD\_ISSET (mouse\_fd, &rfds) ) {                 if ( mouse\_update() ) {                     int x,y,buttons;                     mouse\_getxy(&x, &y);                     buttons = mouse\_getbutton();                     printf("[mouse] [%04d,%04d,%01d]\n", x, y, buttons);                 }             }              if (kbd\_fd >=0 && FD\_ISSET (kbd\_fd, &rfds) ) {                 if ( read\_key() ) {                     const char \*kbd = keyboard\_getstate();                     int key\_updated = keyboard\_update() - 1;                     printf("[keyborad] [%02d,%d]\n",                              key\_updated,                             kbd[key\_updated]);                             // kbd\_data.unicode,                             // kbd\_data.modifiers,                             // kbd\_data.press,                             // kbd\_data.repeat);                 }             }         }     } } |