用Raspberry PI 4 装 Linux ubuntu MATE 运行Mitif 编译

William June 22, 2022

摘要

Ubuntu Linux原本就支持基本X-Window Motif. 它还提供Raspberry PI 4版本, 非常容易下载和装到Raspberry PI 4上. 装完以后只需要两条命令加上 motif development package 就可以编译Motif程序.

图1. Ubuntu MATE 可在Raspberry PI 3,4上运行

I. 下载Ubuntu MATE

下载Ubuntu MATE img文件

https://ubuntu-mate.org/raspberry-pi/

https://ubuntu-mate.org/raspberry-pi/download/

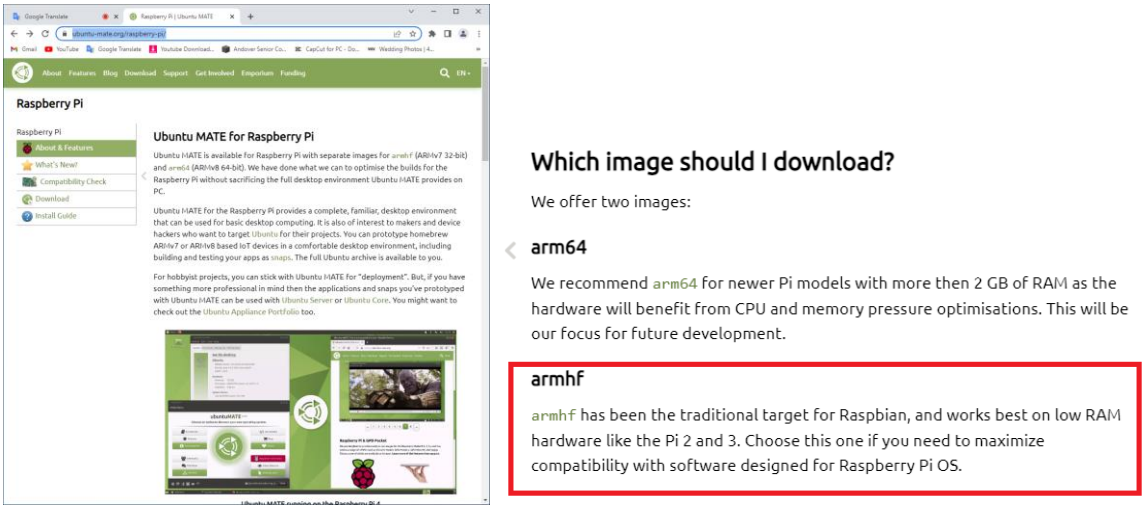


图2. 下载 Ubuntu MATE armhf img文件

这里有两个版本，arm64，armhf．我选择的版本是armhf．这是一个简易版，主机运行的时候比较轻松．

II．在 Raspberry PI 4上装Ubuntu MATE

在 Raspberry PI 4上装Ubuntu MATE非常简单,只要下载一个免费的balenaEtcher程序就可以把下载的Image文件制成Raspberry PI 4的操作系统．

2.1下载 balenaEtcher

https://www.balena.io/etcher/

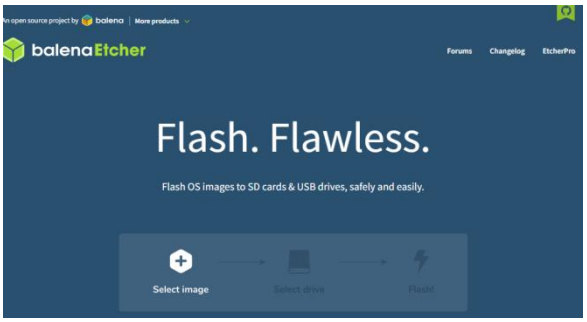


图3． 用 balenaEtcher 把 MATE armhf img 文件装到Micro SD卡上

2.2 用 balenaEtcher 把 MATE armhf img文件装到Micro SD卡上

装完以后，把Micro SD卡插到Raspberry PI 4，boot．这时软件就会进行初始化.回答几个简单的问题以后，reboot．这时就可以看到图3的情况.

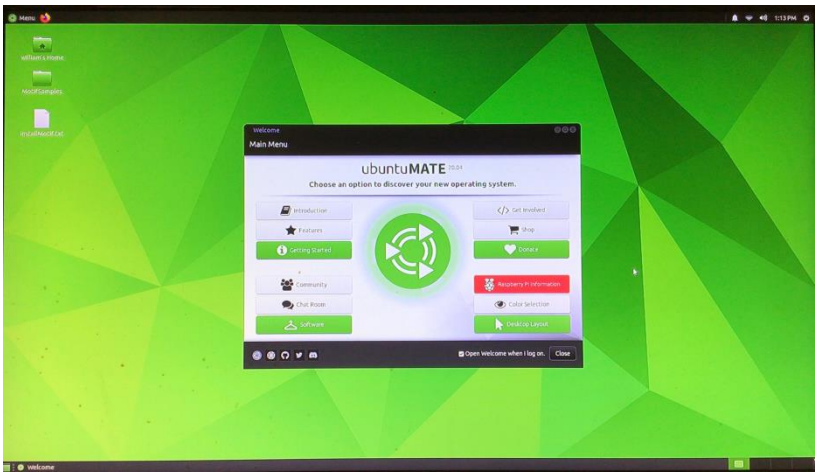


图4． Raspberry PI 4装完以后就可以看到该图像

2.3 装 Motif Development Package

到此, 基本软件已经装完, 但还需要装Motif Development Package这样才能编译自己的应用程序.

1) 启动一个Terminal来运行增加 Package 命令

Manu->SystemTools->Terminal

2) 键入如下命令:

sudo apt install libmotif-dev <CR> (增加 Xm library)

sudo apt install libxmu-dev <CR> (增加 Xmu, Xmuu libraries)

3) 到此,编译Motif的所有软件都已经装载完毕. 可以用以下命令测试一下.

```
cc  -o hello hello.c  -lXm  -lXt  -lX11 <CR>
./hello <CR>
```

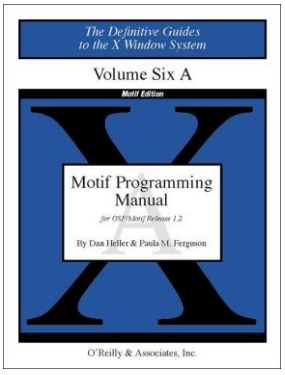
如果一切正常就可以看到图5.

图5. Hello.c

简单测试程序 **hello.c**

```
/* hello.c -- initialize the toolkit using an application context and a
 * toplevel shell widget, then create a pushbutton that says Hello using
 * the varargs interface.
 */
#include <Xm/PushB.h>

main(argc, argv)
int argc;
char *argv[];
{
    Widget        toplevel, button;
    XtAppContext  app;
```

```
    void          button_pushed();
    XmString      label;

    XtSetLanguageProc (NULL, NULL, NULL);

    toplevel = XtVaAppInitialize (&app, "Hello", NULL, 0,
        &argc, argv, NULL, NULL);

    label = XmStringCreateLocalized ("Push here to say hello");
    button = XtVaCreateManagedWidget ("pushme",
        xmPushButtonWidgetClass, toplevel,
        XmNlabelString, label,
        NULL);
    XmStringFree (label);
    XtAddCallback (button, XmNactivateCallback, button_pushed, NULL);

    XtRealizeWidget (toplevel);
    XtAppMainLoop (app);
}

void
button_pushed(widget, client_data, call_data)
Widget widget;
XtPointer client_data;
XtPointer call_data;
{
    printf ("Hello Yourself!0);
}
```



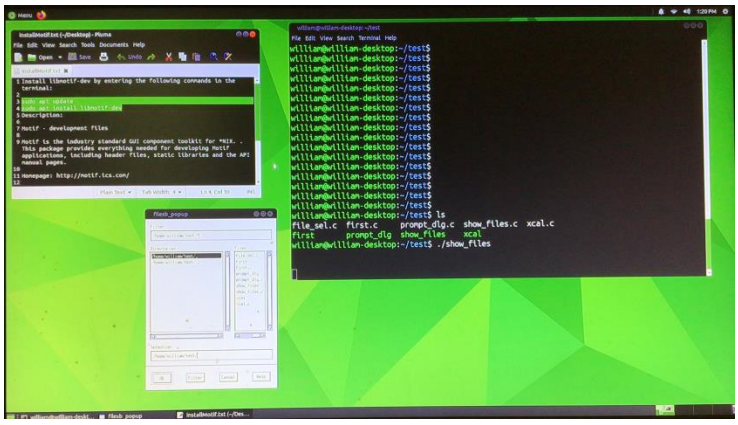


图6. 样例程序 file_sel.c

4) 图5表示了另一个比较复杂的样例程序, file_sel.c

file_sel.c 是官方的编程手册 (Motif Programming Manual)中的一个例子.现在抄录如下.

```
/* file_sel.c -- file selection dialog displays a list of all the writable
```

```
 * files in the directory described by the XmNmask of the dialog.
 * This program demonstrates how to use the XmNfileSearchProc for
 * file selection dialog widgets.
 */
#include <stdio.h>
#include <Xm/Xm.h>
#include <Xm/FileSB.h>
#include <Xm/DialogS.h>
#include <Xm/PushBG.h>
#include <Xm/PushB.h>
#include <X11/Xos.h>
#include <sys/stat.h>

void do_search(), new_file_cb();

/* routine to determine if a file is accessible, a directory,
 * or writable.  Return -1 on all errors or if the file is not
 * writable.  Return 0 if it's a directory or 1 if it's a plain
 * writable file.
 */
int
is_writable(file)
char *file;
{
    struct stat s_buf;

    /* if file can't be accessed (via stat()) return. */
    if (stat (file, &s_buf) == -1)
        return -1;
    else if ((s_buf.st_mode & S_IFMT) == S_IFDIR)
        return 0; /* a directory */
    else if (!(s_buf.st_mode & S_IFREG) || access (file, W_OK) == -1)
        /* not a normal file or it is not writable */
        return -1;
    /* legitimate file */
    return 1;
}

/* main() -- create a FileSelectionDialog */
main(argc, argv)
int argc;
char *argv[];
{
    Widget toplevel, dialog;
    XtAppContext app;
    extern void exit();
    Arg args[5];
    int n = 0;

    XtSetLanguageProc (NULL, NULL, NULL);

    toplevel = XtVaAppInitialize (&app, "Demos",
        NULL, 0, &argc, argv, NULL, NULL);

    XtSetArg (args[n], XmNfileSearchProc, do_search); n++;
    dialog = XmCreateFileSelectionDialog (toplevel, "Files", args, n);
    XtSetSensitive (
```

```
        XmFileSelectionBoxGetChild (dialog, XmDIALOG_HELP_BUTTON), False);
    /* if user presses OK button, call new_file_cb() */
    XtAddCallback (dialog, XmNokCallback, new_file_cb, NULL);
    /* if user presses Cancel button, exit program */
    XtAddCallback (dialog, XmNcancelCallback, exit, NULL);

    XtManageChild (dialog);

    XtAppMainLoop (app);
}

/* a new file was selected -- check to see if it's readable and not
 * a directory.  If it's not readable, report an error.  If it's a
 * directory, scan it just as tho the user had typed it in the mask
 * Text field and selected "Search".
 */
void
new_file_cb(widget, client_data, call_data)
Widget widget;
XtPointer client_data;
XtPointer call_data;
{
    char *file;
    XmFileSelectionBoxCallbackStruct *cbs =
        (XmFileSelectionBoxCallbackStruct *) call_data;

    /* get the string typed in the text field in char * format */
    if (!XmStringGetLtoR (cbs->value, XmFONTLIST_DEFAULT_TAG, &file))
        return;
    if (*file != '/') {
        /* if it's not a directory, determine the full pathname
         * of the selection by concatenating it to the "dir" part
         */
        char *dir, *newfile;
        if (XmStringGetLtoR (cbs->dir, XmFONTLIST_DEFAULT_TAG, &dir)) {
            newfile = XtMalloc (strlen (dir) + 1 + strlen (file) + 1);
            sprintf (newfile, "%s/%s", dir, file);
            XtFree( file);
            XtFree (dir);
            file = newfile;
        }
    }
    switch (is_writable (file)) {
        case 1 :
            puts (file); /* or do anything you want */
            break;
        case 0 : {
            /* a directory was selected, scan it */
            XmString str = XmStringCreateLocalized (file);
            XmFileSelectionDoSearch (widget, str);
            XmStringFree (str);
            break;
        }
        case -1 :
            /* a system error on this file */
            perror (file);
    }
```

```
    XtFree (file);
}

/* do_search() -- scan a directory and report only those files that
 * are writable.  Here, we let the shell expand the (possible)
 * wildcards and return a directory listing by using popen().
 * A *real* application should -not- do this; it should use the
 * system's directory routines: opendir(), readdir() and closedir().
 */
void
do_search(widget, search_data)
Widget widget; /* file selection box widget */
XtPointer search_data;
{
    char          *mask, buf[BUFSIZ], *p;
    XmString       names[256]; /* maximum of 256 files in dir */
    int            i = 0;
    FILE          *pp, *popen();
    XmFileSelectionBoxCallbackStruct *cbs =
        (XmFileSelectionBoxCallbackStruct *) search_data;

    if (!XmStringGetLtoR (cbs->mask, XmFONTLIST_DEFAULT_TAG, &mask))
        return; /* can't do anything */

    sprintf (buf, "/bin/ls %s", mask);
    XtFree (mask);
    /* let the shell read the directory and expand the filenames */
    if (!(pp = popen (buf, "r")))
        return;
    /* read output from popen() -- this will be the list of files */
    while (fgets (buf, sizeof buf, pp)) {
        if (p = index (buf, '0))
            *p = 0;
        /* only list files that are writable and not directories */
        if (is_writable (buf) == 1 &&
            (names[i] = XmStringCreateLocalized (buf)))
            i++;
    }
    pclose (pp);
    if (i) {
        XtVaSetValues (widget,
            XmNfileListItems,      names,
            XmNfileListItemCount,  i,
            XmNdirSpec,            names[0],
            XmNlistUpdated,        True,
            NULL);
        while (i > 0)
            XmStringFree (names[--i]);
    } else
        XtVaSetValues (widget,
            XmNfileListItems,      NULL,
            XmNfileListItemCount,  0,
            XmNlistUpdated,        True,
            NULL);
}
```