Purple Pi R1 文件系统配置

SD卡根文件系统

- buildroot
- QT 5.15
  - TSLIB 源码编译
  - QT 源码编译
  - 验证
- Ubuntu
  - 1.获取资源
  - 2.搭建虚拟机环境
  - 3.添加自己需要的服务和工具
  - 4.制作文件系统启动卡
  - 5.将文件系统打包并挂载到SD卡

OpenWRT

Industio 触觉智能

Purple Pi R1

文件系统配置

SD卡根文件系统

一般情况下，我们使用nand/nor flash，容量一般为128/256/512Mbit，有时候我们需要一个更大的空间，使得能够存放更多文件。而SD卡的容量足够大（以GB为单位），可以满足上面的需求。

做法是将uboot和kernel放置在flash中，然后把我们较大的根文件系统放置在SD卡里面。之所以不能把uboot和kernel也放置在SD卡里，是因为SSD20X仅支持从flash启动。

buildroot

- **制作SD卡文件系统**

Plain Text | 复制代码

```
industio@industio$:mkdir sd_rootfs
industio@industio$:cd sd_rootfs
industio@industio$:cp ../project/image/output/rootfs/* ./ -rf
industio@industio$:industio@industio$:cp ../project/image/output/customer/ . -rf
industio@industio$:cp ../project/image/output/appconfigs/ . -rf
industio@industio$:cp ../project/image/output/miservice/config/ . -rf
industio@industio$:tar -cvf rootfs.tar ./*
industio@industio$:touch make_sd_rootfs.sh
```

make_sd_rootfs.sh内容如下：

```
#!/bin/sh
PWD=$(pwd)
images_dir=${PWD}/images_for_mksdcard
if [ "$1" == "" ]; then
echo "!!!!!!!!!!!! ./make_sd_rootfs.sh /dev/sdb !!!!!!!!!!!"
exit 0
fi
sfdisk ${1}
mkfs.ext3 -F -j ${1}1
mkdir tmp_rootfs
mount -t ext3 ${1}1 tmp_rootfs
tar -xvf ./rootfs.tar -C tmp_rootfs
umount tmp_rootfs
rm -rf tmp_rootfs
```

```
industio@industio$:chmod a+x make_sd_rootfs.sh
```

把SD卡接入到ubuntu虚拟机中，首先要把SD umount掉，假设被是识别为/dev/sdb，则执行：

```
industio@industio$:umount /dev/sd*
industio@industio$:sudo ./make_sd_rootfs.sh /dev/sdb
```

等待制作完成。

```
industio@industio:~/ssd20x/sd_rootfs$ sudo ./make_sd_rootfs.sh /dev/sdb
Checking that no-one is using this disk right now ... OK

Disk /dev/sdb: 29.1 GiB, 31266439168 bytes, 61067264 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0b8b64dd

Old situation:

Device     Boot Start      End  Sectors  Size Id Type
/dev/sdb1        2048 61067263 61065216 29.1G 83 Linux

>>> Created a new DOS disklabel with disk identifier 0x244dfeac.
/dev/sdb1: Created a new partition 1 of type 'Linux' and of size 29.1 GiB.
/dev/sdb2: Done.
```

本地验证：

```
industio@industio$:sudo mount /dev/sdb1 /mnt
industio@industio$:sudo umount /mnt
```

制作完成，将SD卡取出并接入到开发板的SD卡座上。

- **kernel支持EXT2/3/4**

需要确保kernel支持EXT2/3/4

```
industio@industio$:cd kernel
industio@industio$:ARCH=arm make menuconfig
```

File systems —>

<*> The Extended 3 (ext3) filesystem

如果没有支持，需重新配置并更新kernel。

保存当前config：

```
industio@industio$:cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig -f
```

- **设置bootargs**

uboot模式下执行：

```
SigmaStar # setenv bootargs console=ttyS0,115200 root=/dev/mmcblk1p1 rw rootwait=1 LX_MEM=0x3f00000 mma_heap=mma_heap_name0,miu=0,sz=0xa00000 mma_memblock_remove=1 highres=off mmap_reserved=fb,miu=0,sz=0x300000,max_start_off=0x3300000,max_end_off=0x3600000 mtdparts=nand0:384k@1280k(IPL0),384k(IPL1),384k(IPL_CUST0),384k(IPL_CUST1),768k(UBOOT0),768k(UBOOT1),256k(ENV),256k(ENV1),0x20000(KEY_CUST),0x60000(LOGO),0x500000(KERNEL),0x500000(RECOVERY),-(UBI)

SigmaStar # saveenv
```

- **验证**

重新上电后，可以看到SD卡中的文件系统已经被正确挂载了，并且空间足够大。

```
Registering SWP/SWPB emulation handler
ms_rtcpwc 1f006800.rtcpwc: setting system clock to 1970-01-01 01:11:02 UTC (4262)
OF: fdt:not creating '/sys/firmware/fdt': CRC check failed
EXT4-fs (mmcblk0p1): mounting ext3 file system using the ext4 subsystem
EXT4-fs (mmcblk0p1): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext3 filesystem) on device 179:1.
devtmpfs: mounted
This architecture does not have kernel memory protection.
random: fast init done
EXT4-fs (mmcblk0p1): re-mounted. Opts: data=ordered
```

```
#mount
```

```
# mount
/dev/root on / type ext3 (rw,relatime,data=ordered)
devtmpfs on /dev type devtmpfs (rw,relatime,size=21352k,nr_inodes=5338,mode=755)
proc on /proc type proc (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620,ptmxmode=666)
tmpfs on /dev/shm type tmpfs (rw,relatime,mode=777)
tmpfs on /tmp type tmpfs (rw,relatime)
tmpfs on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
sysfs on /sys type sysfs (rw,relatime)
/dev/mmcblk0p1 on /sdcard type ext3 (rw,relatime,data=ordered)
mdev on /dev type tmpfs (rw,relatime)
none on /sys/kernel/debug type debugfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
```

```
#df -h
```

```
# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                28.5G     82.5M     27.0G   0% /
devtmpfs                 21.9M         0     21.9M   0% /dev
df: /dev/shm: No such file or directory
tmpfs                    21.9M     48.0K     21.8M   0% /tmp
tmpfs                    21.9M     28.0K     21.8M   0% /run
/dev/mmcblk0p1           28.5G     82.5M     27.0G   0% /sdcard
mdev                     21.9M         0     21.9M   0% /dev
#
```

QT 5.15

TSLIB 源码编译

由于QT依赖TSLIB，因此在编译QT源码前先编译TSLIB。

从 https://github.com/libts/tslib/releases/tag/1.15 中下载 tslib-1.15tar.bz2 到Ubuntu 虚拟机下并解压，进入解压目录，新建install目录：

Plain Text | 复制代码

```
1   industio@industio$:cd tslib-1.15
2   industio@industio$:mkdir install
```

```
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$ ls
acinclude.m4  ChangeLog     config.sub  depcomp  INSTALL     Makefile.am  plugins  tests
aclocal.m4    compile       configure   doc      install-sh  Makefile.in  README   THANKS
AUTHORS       config.guess  configure.ac  etc    ltmain.sh   missing      README.md  tools
autogen.sh    config.h.in   COPYING     install  m4          NEWS         src      tslib.pc.in
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$
```

确认交叉编译链是否匹配：

```
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$ which arm-linux-gnueabihf-gcc
/home/ronnie/work/ssd201/ssd20x_sdk_v009/gcc-arm-8.2-2018.08-x86_64-arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$
```

获取install目录的完整路径：

```
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$ pwd
/home/ronnie/work/ssd201/qt/tslib-1.15
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$
```

开始交叉编译：

Plain Text | 复制代码

```
industio@industio$:./configure --prefix=/home/ronnie/work/ssd201/qt/tslib-1.15/install --host=arm-linux-gnueabihf
```

```
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$ ./configure --prefix=/home/ronnie/work/ssd201/qt/tslib-1.15 -host=arm-linux-gnueabihf
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for arm-linux-gnueabihf-strip... arm-linux-gnueabihf-strip
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether make supports nested variables... (cached) yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... arm-unknown-linux-gnueabihf
checking for arm-linux-gnueabihf-gcc... arm-linux-gnueabihf-gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... yes
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether arm-linux-gnueabihf-gcc accepts -g... yes
checking for arm-linux-gnueabihf-gcc option to accept ISO C89... none needed
checking whether arm-linux-gnueabihf-gcc understands -c and -o together... yes
checking for style of include used by make... GNU
checking dependency style of arm-linux-gnueabihf-gcc... none
checking how to run the C preprocessor... arm-linux-gnueabihf-gcc -E
checking whether the C compiler supports -fvisibility=hidden... yes
```

Plain Text | 复制代码

```
industio@industio$: make
industio@industio$: make install
```

```
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$ ls install
bin  etc  include  lib  share
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$ file install/bin/ts_calibrate
install/bin/ts_calibrate: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, i
nterpreter /lib/ld-, for GNU/Linux 3.2.0, not stripped
ronnie@wt_rd_server:~/work/ssd201/qt/tslib-1.15$
```

QT 源码编译

从 http://download.qt.io/archive/qt/5.15/5.15.0/single/ 下载 qt-everywhere-src-5.15.0.tar.xz 到Linux系统下并解压;

```
ronnie@wt_rd_server:~/work/ssd201/qt$ ls
qt-everywhere-src-5.15.0  qt-everywhere-src-5.15.0.tar  tslib-1.15
ronnie@wt_rd_server:~/work/ssd201/qt$
```

进入解压目录，然后新建编译脚本 make.sh

```
1  industio@industio$: cd qt-everywhere-src-5.15.0
2  industio@industio$:industio@industio$: touch make.sh
3  industio@industio$: chmod 777 make.sh
```

```
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$ ls
_clang-format   install                    QT_demo_pieces_rar.rar  qtsensors
clocks          LICENSE.FDL                qtdoc                   qtserialbus
coin            LICENSE.GPLv2              qtgamepad               qtserialport
config.cache    LICENSE.GPLv3              qtgraphicaleffects      qtspeech
config.log      LICENSE.LGPLv21            qtimageformats          qtsvg
config.opt      LICENSE.LGPLv3             qtlocation              qttools
config.status   LICENSE.QT-LICENSE-AGREEMENT qtlottie              qttranslations
config.summary  Makefile                   qtmacextras             qtvirtualkeyboard
config.tests    make.sh                    qtmultimedia            qtwayland
configure       qt3d                       qtnetworkauth           qtwebchannel
configure.bat   QT_5.15.0_install.tar      qt.pro                  qtwebengine
configure.json  qtactiveqt                 qtpurchasing            qtwebglplugin
demos           qtandroidextras            qtquick3d               qtwebsockets
examples        qtbase                     qtquickcontrols         qtwebview
examples.tar    qtcharts                   qtquickcontrols2        qtwinextras
gnuwin32        qtconnectivity             qtquicktimeline         qtx11extras
hello           qtdatavis3d                qtremoteobjects         qtxmlpatterns
hello_new       qtdeclarative              qtscript                README
hello_world     QT_demo_pieces_rar         qtscxml                 shared
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$
```

修改qmake.conf

```
1  industio@industio$:vi qtbase/mkspecs/linux-arm-gnueabi-g++/qmake.conf
```

```
#

MAKEFILE_GENERATOR      = UNIX
CONFIG                 += incremental
QMAKE_INCREMENTAL_STYLE = sublib

include(../common/linux.conf)
include(../common/gcc-base-unix.conf)
include(../common/g++-unix.conf)

QT_QPA_PLATFORM=linuxfb:fb=/dev/fb0:rotation=0

# modifications to g++.conf
QMAKE_CC                = arm-linux-gnueabihf-gcc
QMAKE_CXX               = arm-linux-gnueabihf-g++
QMAKE_LINK              = arm-linux-gnueabihf-g++
QMAKE_LINK_SHLIB        = arm-linux-gnueabihf-g++

# modifications to linux.conf
QMAKE_AR                = arm-linux-gnueabihf-ar cqs
QMAKE_OBJCOPY           = arm-linux-gnueabihf-objcopy
QMAKE_NM                = arm-linux-gnueabihf-nm -P
QMAKE_STRIP             = arm-linux-gnueabihf-strip
load(qt_config)
~
~
~
```

添加编译脚本

```
1   industio@industio$: vi make.sh
```

make.sh内容如下：

make.sh

Plain Text | 复制代码

```
#!/bin/sh
PWD=`pwd`
  mkdir install
  ./configure \
  -prefix $PWD/install \
  -static \
  -release \
  -opensource \
  -xplatform linux-arm-gnueabi-g++ \
  -optimized-qmake -pch \
  -qt-libjpeg \
  -qt-libpng \
  -qt-zlib \
  -no-opengl \
  -skip qt3d \
  -skip qtcanvas3d \
  -skip qtpurchasing \
  -skip qtlocation \
  -skip qttools \
  -no-sse2 \
  -no-openssl \
  -no-cups \
  -no-glib \
  -no-iconv \
  -tslib \
  -linuxfb \
  -I /home/lck/ssd20x/qt/tslib-1.15/install/include \
  -L /home/lck/ssd20x/qt/tslib-1.15/install/lib \
  -recheck-all \
  -make examples

  make -j16
  make install
```

开始交叉编译：

Plain Text | 复制代码

```
industio@industio$: ./make.sh
```

等待一段时间后，编译完成：

```
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$ ls install/
bin  disp_init  doc  include  lib  mkspecs  plugins  qml
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$ ls install/bin/
canbusutil        qlalr        qmlimportscanner  qmltyperegistrar  syncqt.pl
fixqt4headers.pl  qmake        qmllint           qscxmlc           tracegen
moc               qml          qmlmin            qvkgen            uic
qdbuscpp2xml      qmlcachegen  qmlpreview        rcc               xmlpatterns
qdbusxml2cpp      qmlformat    qmltestrunner     repc              xmlpatternsvalidator
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$ file install/bin/qml
install/bin/qml: ELF 32-bit LSB executable, ARM, EABI5 version 1 (GNU/Linux), dynamically linked, inter
preter /lib/ld-, for GNU/Linux 3.2.0, stripped
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$
```

设置qmake环境变量：

Plain Text | 复制代码

```
industio@industio$: vi ~/.bashrc
 export PATH=/home/ronnie/work/ssd201/qt/qt-everywhere-src-5.15.0/install/bin:$PATH
```

```
export PATH=/home/ronnie/work/ssd201/ssd20x_sdk_v4.0.0/gcc-arm-9.2-2019.08-x86_64-arm-linux-gnueabihf/bin:$PATH

export PATH=/home/ronnie/work/ssd201/qt/qt-everywhere-src-5.15.0/install/bin:$PATH
```

Plain Text | 复制代码

```
industio@industio$: source ~/.bashrc
industio@industio$: which qmake
```

```
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$ source ~/.bashrc
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$ which qmake
/home/ronnie/work/ssd201/qt/qt-everywhere-src-5.15.0/install/bin/qmake
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0$
```

验证

通过 QTCreator 新建工程 hello，然后拷贝到 Linux 系统下：

```
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$ ls
main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  ui_mainwindow.h
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$
```

在hello目录下，执行 qmake –project，生成 hello.pro：

```
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$ qmake -project
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$ ls
hello.pro  main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  ui_mainwindow.h
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$
```

编辑hello.pro，作以下修改：

```
#############################################################################
# Automatically generated by qmake (3.1) Wed Aug 12 07:48:36 2020
#############################################################################
QT       += core gui widgets

TEMPLATE = app
TARGET = hello
INCLUDEPATH += .

# You can make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# Please consult the documentation of the deprecated API in order to know
# how to port your code away from it.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

# Input
HEADERS += mainwindow.h ui_mainwindow.h
FORMS += mainwindow.ui
SOURCES += main.cpp mainwindow.cpp

QTPLUGIN        += qlinuxfb
```

确保交叉编译链环境配置正确：

```
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$ which arm-linux-gnueabihf-gcc
/home/ronnie/work/ssd201/sdk/toolchain/gcc-arm-8.2-2018.08-x86_64-arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$
```

编译工程

Plain Text | 复制代码

```
industio@industio$: qmake
industio@industio$: ARCH=arm make
```

编译完成后，将在目录下生成 hello 可执行文件：

```
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$ ls
hello                  hello.pro  mainwindow.cpp  mainwindow.ui       moc_mainwindow.o
hello_plugin_import.cpp  main.cpp   mainwindow.h    Makefile            moc_predefs.h
hello_plugin_import.o    main.o     mainwindow.o    moc_mainwindow.cpp  ui_mainwindow.h
ronnie@wt_rd_server:~/work/ssd201/qt/qt-everywhere-src-5.15.0/hello$
```

将 libts.so*拷贝到板子的 /usr/lib 目录下：

```
/root # ls /usr/lib/libts.so*
/usr/lib/libts.so        /usr/lib/libts.so.0      /usr/lib/libts.so.0.9.0
/root #
/root #
```

拷贝字库到 /usr/lib/font 目录下，字库可以从 test/qt/font/中获得：

```
/root # ls /usr/lib/font/
fzcircle.ttf
/root #
```

添加QT环境变量：

Plain Text | 复制代码

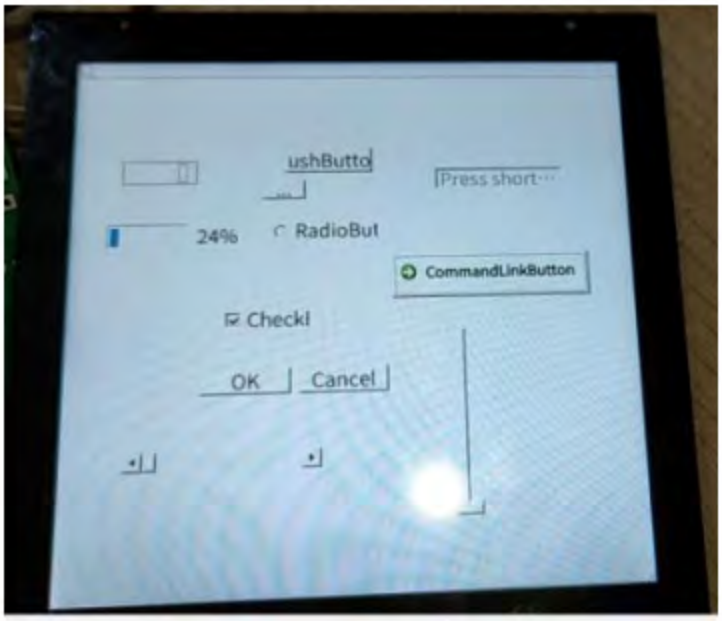
```
industio@industio$: vi /etc/profile
```

```
#QT env
export TSLIB_PLUGINDIR=/qt/ts
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_CONFFILE=/ts/ts.conf
export TSLIB_TSDEVICE=/dev/input/event0
export TSLIB_CALIBFILE=/etc/pointercal
export QT_QPA_PLATFORM=linuxfb
export QT_QPA_FONTDIR=/usr/lib/font
```

将 disp_init和 hello 拷贝到板子上，先在后台运行 disp_init，再运行hello：

Plain Text | 复制代码

```
# ./disp_init &
# ./hello
```



Ubuntu

1.获取资源

我们已经制作好的Ubuntu的文件系统：
git clone https://github.com/industio/ido-ubuntu-base-20.04.git

2.搭建虚拟机环境

将制作好的文件系统下载到自己的虚拟机中。

修改tmp目录权限为 777，在apt update的时候会在tmp目录下创建一些临时文件，所以要给tmp目录写权限。

Plain Text | 复制代码

```
industio@industio$:chmod 777  ssd20x/ido-ubuntu-base-20.04/tmp
```

修改resolv.conf文件,添加dns。

Plain Text | 复制代码

```
industio@industio$:vim ssd20x/ubuntu_base/etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

3.添加自己需要的服务和工具

需要先将base文件系统挂载到虚拟机下

- 编写挂载脚本

在解压目录的上一级目录下新建一个ms.sh文件，文件内容如下，并赋予文件777的权限。
这里我们提供的文件系统带有了无需添加。

ms.sh Plain Text 复制代码

```
#!/bin/bash
mnt ()
{
    echo "MOUNTING"
    sudo mount -t proc /proc ${2}proc
    sudo mount -t sysfs /sys ${2}sys
    sudo mount -o bind /dev ${2}dev
    sudo mount -o bind /dev/pts ${2}dev/pts
    sudo chroot ${2}
}
umnt ()
{
    echo "UNMOUNTING"
    sudo umount ${2}proc
    sudo umount ${2}sys
    sudo umount ${2}dev/pts
    sudo umount ${2}dev
}

if [ "$1" = "-m" ] && [ -n "$2" ];
then
    mnt $1 $2
    echo "mnt -m pwd"
elif [ "$1" = "-u" ] && [ -n "$2" ];
then
    umnt $1 $2
    echo "mnt -u pwd"
else
    echo ""
    echo "Either 1'st, 2'nd or bothparameters were missing"
    echo ""
    echo "1'st parameter can be one ofthese: -m(mount) OR -u(umount)"
    echo "2'nd parameter is the full pathof rootfs directory(with trailin
g '/')"
    echo ""
    echo "For example: ch-mount -m/media/sdcard/"
    echo ""
    echo 1st parameter : ${1}
    echo 2nd parameter : ${2}
fi
```

- 挂载

```
industio@industio$:sudo ./ms.sh -m home/xxxx/ssd20x/ubuntu_base/
```

- 卸载

```
industio@industio$:sudo ./ms.sh -u home/xxxx/ssd20x/ubuntu_base/
```

模拟root也可以使用chroot命令替代。
成功挂载后就可以添加自己需要的工具和服务了。

```
industio@industio$:apt update
industio@industio$:apt install usbutils
industio@industio$:apt install sudo
industio@industio$:apt install language-pack-en-base
industio@industio$:apt install ssh
industio@industio$:apt install net-tools
industio@industio$:apt install ethtool
industio@industio$:apt install ifupdown
industio@industio$:apt install iputils-ping
industio@industio$:apt install rsyslog
industio@industio$:apt install htop
industio@industio$:apt install vi
industio@industio$:apt install dhcpcd5
industio@industio$:apt install samba samba-common
industio@industio$:apt install wpasupplicant
industio@industio$:apt install jq
industio@industio$:apt install alsa-base
industio@industio$:apt install minicom
```

到此输入 **exit** 退出挂载界面，并卸载文件系统。

```
industio@industio$:sudo ./ms.sh -u home/xxxx/ssd20x/ubuntu_base/
```

注意事项：

（1）每次退出挂载在虚拟机的base文件系统，需要执行卸载指令，否则二次挂载会出问题。

4.制作文件系统启动卡

创建一个SD卡制作脚本，这里我们已经做好了可以直接使用 。

Plain Text 复制代码

```
industio@industio$:touch make_ubuntu_rootfs.sh
```

make_sd_rootfs.sh内容如下：

Plain Text 复制代码

```
#!/bin/sh
PWD=$(pwd)
images_dir=${PWD}/images_for_mksdcard
if [ "$1" == "" ]; then
echo "!!!!!!!!!!!! ./make_ubuntu_rootfs.sh /dev/sdb !!!!!!!!!!"
exit 0
fi
sfdisk ${1}
mkfs.ext3 -F -j ${1}1
mkdir tmp_rootfs
mount -t ext3 ${1}1 tmp_rootfs
tar -xvf ./rootfs.tar -C tmp_rootfs
umount tmp_rootfs
rm -rf tmp_rootfs
```

Plain Text 复制代码

```
industio@industio$:chmod a+x make_ubuntu_rootfs.sh
```

压缩文件系统

```
industio@industio$:industio@industio$:cd ubuntu_base/
industio@industio$:sudo tar -cvf rootfs.tar ./*
```

5.将文件系统打包并挂载到SD卡

把SD卡接入到ubuntu虚拟机中，首先要把SD umount掉，假设被是识别为/dev/sdb，则执行：

```
industio@industio$:umount /dev/sd*
industio@industio$:sudo ./make_ubuntu_rootfs.sh /dev/sdb
```

等待制作完成。

```
industio@industio:~/ssd20x/sd_rootfs$ sudo ./make_sd_rootfs.sh /dev/sdb
Checking that no-one is using this disk right now ... OK

Disk /dev/sdb: 29.1 GiB, 31266439168 bytes, 61067264 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0b8b64dd

Old situation:

Device     Boot Start      End  Sectors  Size Id Type
/dev/sdb1        2048 61067263 61065216 29.1G 83 Linux

>>> Created a new DOS disklabel with disk identifier 0x244dfeac.
/dev/sdb1: Created a new partition 1 of type 'Linux' and of size 29.1 GiB.
/dev/sdb2: Done.
```

本地验证：

```
industio@industio$:sudo mount /dev/sdb1 /mnt
industio@industio$:sudo umount /mnt
```

```
industio@industio:~/ssd20x/sd_rootfs$ sudo mount /dev/sdb1 /mnt
industio@industio:~/ssd20x/sd_rootfs$ ls /mnt/
appconfigs  config    dev       etc  lib32   logo      lost+found  mi_alsa.ko  opt   root  sbin    snd.ko      snd-timer.ko  sys  udisk  var
bin         customer  disp_init  lib  linuxrc  logo.jpg  media       mnt         proc  run   sdcard  snd-pcm.ko  soundcore.ko  tmp  usr    vendor
industio@industio:~/ssd20x/sd_rootfs$
```

制作完成，将SD卡取出并接入到开发板的SD卡座上。

- **kernel支持EXT2/3/4**

需要确保kernel支持EXT2/3/4

```
industio@industio$:cd kernel
industio@industio$:ARCH=arm make menuconfig
```

File systems —>

<*> The Extended 3 (ext3) filesystem

如果没有支持，需重新配置并更新kernel。

保存当前config：

```
industio@industio$:cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig -f
```

- **设置bootargs**

uboot模式下执行：

```
SigmaStar # setenv bootargs console=ttyS0,115200 root=/dev/mmcblk1p1 rw rootwait=1 LX_MEM=0x3f00000 mma_heap=mma_heap_name0,miu=0,sz=0xa00000 mma_memblock_remove=1 highres=off mmap_reserved=fb,miu=0,sz=0x300000,max_start_off=0x3300000,max_end_off=0x3600000 mtdparts=nand0:384k@1280k(IPL0),384k(IPL1),384k(IPL_CUST0),384k(IPL_CUST1),768k(UBOOT0),768k(UBOOT1),256k(ENV),256k(ENV1),0x20000(KEY_CUST),0x60000(LOGO),0x500000(KERNEL),0x500000(RECOVERY),-(UBI)

SigmaStar # saveenv
```

重启就可以看到使用的是SD卡中的Ubuntu-base的文件系统了。

OpenWRT

OpenWRT-SDK：https://github.com/wireless-tag-com/openwrt-ssd20x

OpenWRT编译烧录说明： http://doc.8ms.xyz/docs/faq/faq-1cqk9k5td3m9v