

# Pascal 开发方向入手 IOT 的完全攻略

文档版本：2.11

更新日志

2018-10-1: 更新在小内存 IOT 主板修改交换文件指引

2018-9-23: 更新不同 Linux 版本的 fpc 编译器下载指引

2018-9-22: 早期版本 Lazarus 编译时只能认 fpc3.0.4, 本次更新解决了来自 github 最新的 fpc 与 lazarus 源代码的编译兼容性问题

目录	
概述 .....	3
硬件选购.....	3
机电系统.....	3
外置电源，或移动电源模块.....	3
传感器 .....	3
IOT 第一次上手 .....	4
第一步，准备工作.....	4
第二步，安装.....	5
第三步，成功.....	5
第四步，验证硬件参数.....	6
初始化 IOT 中的 Linux.....	6
安装 lazarus 所需要的重要依赖包 .....	7
安装和卸载原有系统的 Lazarus+FPC(首次安装系统可略过) .....	7
修改内存交换文件(小内存系统，如 512M).....	8
安装附属小工具(此步骤可略过) .....	8
安装 gcc(此步骤可略过).....	8
把 freepascal 和 lazarus 复制到 IOT 设备中 .....	9
我们在树莓派的在文件管理器找到我们刚才传的 fpc 相关包 .....	10
在 root 权限复制和解压.....	10
使用自编译方式：编译 fpc 编译器.....	11
验证编译器是否支持自编译.....	14
FPC 编译器所支持的 ARM 处理器 Model .....	16
编译 fpc(编译大规模部署版本 3.0.4) .....	17
编译 fpc(来自 github 的当天更新版本).....	18
编译和安装 Lazarus 的 IDE .....	19
fpc 编译时报错：找不到 crtbegin.o .....	21
Freepascal 的 IOT 通讯组件 .....	23
关于 fpc+lazarus 构建中文指南作者 .....	23

# 概述

本篇攻略详尽介绍了 Pascal 流派从 IOT 硬件选购，到 diy 出操作系统，再到 diy 出开发环境的全部过程。我们一起跟上万物互联的大潮吧。

## 硬件选购

强烈建议使用 ubuntu 系统体系的 IOT 硬件，凡是下列网站罗列出来的硬件平台，你都可以选择购买，价格从几十到几百块之间。如果你是切入 IOT 的新人，建议购买套机。

<https://cn.ubuntu.com/internet-of-things/>

除此之外，sd 卡读卡器，hdmi，dvi，基础设备在购买之前也要准备好  
我个人建议购买树莓派 3b+，它更适合新人，购买之前认准英国原版

如果是树莓派 3，SD 卡要选 16G 或则 32G 以上的容量。

总而言之，你需要了解到 IOT 硬件开发商，自己家 diy 的 Linux 系统需求，这点很重要，多看官网 wiki 和技术 paper。

如果你是高通，三星的 IOT 用户，选购能组合的硬件时，也是如此。

## 机电系统

如果你的开发涉及到机电芯一体化系统，请不要来问我，因为机电模块太多了，我不知道怎么给你推荐。自己去解决吧。

## 外置电源，或移动电源模块

直接在某宝上购买即可

## 传感器

请直接联系代理商，或则某宝搜索

# IOT 第一次上手

初次收货时，代理商会发给你 sd 卡，主板，风扇，电源这些基本零件。不要指望代理商能教会你 diy 出系统使用，他们只是代理而已。

## 第一步，准备工作

SD 卡读卡器，类似多功能 U 盘，有很多插口，SD 插入就变成 U 盘

IOT 的操作系统镜像包往 Ubuntu 方向靠，FPC 支持良好，<http://www.ubuntu.com>

SD 卡格式软件，<https://www.sdcard.org>

SD 卡烧录软件，<https://sourceforge.net/projects/win32diskimager/>

FreePascal 编译器源代码的大规模部署版本(我们用于自编译器的引导编译，注意：撰写本文时 [freepascal.org](http://freepascal.org) 官网没有提供支持 linux arm aarch64 架构的自编译引导程序下载，[安装 IOT 的操作系统勿选基于 AARCH64 架构的 LINUX 系统](#)，根据 IOT 操作系统和处理器架构选择对应的编译器下载)

<https://www.freepascal.org/download.var>

最新的 fpc 源码在这里下(我们需要 diy 出来的目标编译器)

<https://github.com/graemeg/freepascal>

lazarus2.0.0rc1 最后的发行版源码，我没有用这个版本，编译失败时可以采用

<https://sourceforge.net/projects/lazarus/files/Lazarus%20Zip%20%20GZip/>

最新的 Lazarus 源码在这里下，<https://github.com/graemeg/lazarus>

远程桌面 realvnc, <https://www.realvnc.com>

Winscp 远程文件传输客户端，<https://winscp.net>

Putty 远程命令行工具，<https://putty.org/>

如果你访问国外网站很慢，还要准备一下自己的代理软件，比如购买一个月硅谷主机，然后用美国主机来下载上面的软件。因为现在 winserver 在国外跑后台都需要正版授权，win 系统的费用大概 100-700 元间，根据实际情况来决定吧。本文不会让直接引用 git 和 svn 源码，网速太慢，不符合国情。

## 第二步，安装

- 先按要求给 IOT 硬件组装风扇，粘贴散热片，组装盒子
- 格式化 SD 卡，然后使用 SD 卡烧录软件将 IOT 镜像包烧录进去
- 将 SD 插入 IOT 硬件，插上键盘，鼠标，显示器，通电启动
- 如果这时，出现无法启动，或则是系统不认键盘鼠标显示器这些外设，恭喜，你入坑了。
  - a) 检查你的 linux 系统镜像是否是硬件开发商所提供
  - b) 如果不是开发商的系统镜像，就去开发商论坛或则 wiki 查找，都有解决办法
  - c) 如果你的系统是开发商自家的，但是又无法启动，很可能就是你安装风扇和散热片时把芯片弄坏了了，去检查一下，如果芯片脱落或则锡焊松动，恭喜，你报废了你的第一个 IOT 设备，一个设备几百块，再买一个吧。

新人在安装 IOT 系统时都有不少问题，记住一个原则：选 IOT 硬件开发商自家的带有 Desktop(gtk2+,gtk4+这类接口)操作系统镜像，直接烧录安装。初次接触，自动化的 desktop 是首选，到以后你有点 Linux 经验，再选择 Core 或则源码版本 Linux 自行 diy 比较好。

经验：国外的网站访问很慢，在安装时，一切和 Update 有关的地方，要全部 skip。

经验：安装前多读官网中有关 wifi，以太网的相关说明，一次性做到安装完成，可以联网（这点太重要了），现在不能联网的系统，就不能算操作系统。

经验：在任何情况下，我们都要安装一个 VNC 和 SSH（可以远程操作）

## 第三步，成功

如果第一步，你准备充分，到第三步只需 10 几分钟，如果缺少操作系统常识，硬件安装过程操作错误，反复弄，装 2, 3 个小时甚至一整天，也很正常。

先不管过程，到这一步，我就认为你已经成功安装了自己的第一个 IOT 系统了

## 第四步，验证硬件参数

中国是代工大国，也意味着，中国人具有复制树莓派设备的能力，这一步在任何时候都很有必要，避免入手到假货

查看 cpu 信息

`cat /proc/cpuinfo`

查看内存

`free -h`

查看磁盘

`lsblk`

`df -hT`

查看更多硬件信息

`dmesg`

查看树莓派型号

`cat /proc/device-tree/model`

查看树莓派系统位数

`getconf LONG_BIT`

查看 usb

`lsusb`

查看其他硬件

`lsmod`

查看 CPU 频率

`vcgencmd get_config arm_freq`

仔细参看这些参数输出的数值，如果不符合代理商给你的参数，就是问题货。

## 初始化 IOT 中的 Linux

我们要打开操作系统中的 VNC 和 SSH 的支持，这里我就不多做介绍了，各个操作系统都有自己的包和配置。自行搜索方法。

最好打开 root 权限，这里我就不多做介绍了，以 Root 登录会省很多事(后面会省 100 条++ 的命令行指令操作)

以上工作都完成后，使用外部设备测试一下，VNC 是否连接成功，SSH 是否连接成功

## 安装 lazarus 所需要的重要依赖包

注意：不要修改更新源，中间也不要中断

使用 Root 权限

```
sudo su
```

现在我们安装 LAZARUS 所需要的依赖包，过程不要断，也不要 CTRL+C，中断非常容易损坏依赖库体系，修复依赖库比重装系统更麻烦，运行这一步，建议插网线跑，保证成功。一旦出问题，就重装一次系统

```
apt-get -y install libx11-dev libgdk-pixbuf2.0-dev libcairo2-dev gir1.2-cogl-pango-1.0 libpangox-1.0-dev xorg-dev libgtk2.0-dev libpango1.0-dev binutils-dev
```

如果只编译 FPC

```
apt-get -y install binutils-dev
```

## 安装和卸载原有系统的 Lazarus+FPC(首次安装系统可略过)

打开控制台

如果你是以 Root 方式登录，下面这句可以省略

```
sudo su
```

从更新源下载你的 Linux 升级包索引（此步骤可略过）

```
apt-get update
```

从升级包索引更新现有的 Linux 组件（此步骤可略过）

```
apt-get upgrade
```

查看 fpc 版本，如果显示的版本是你需要的，那么后面的所有内容可以省略。如果没有找到 fpc，我们就不要再移除了，如果找到了 fpc 编译器，并且这个编译器不符合我们的需求，那么就继续往下走

```
fpc -iW
```

移除 lazarus 的 IDE 包

```
apt-get remove lazarus
```

移除 fpc 编译器包

```
apt-get remove fpc
```

移除垃圾（此步骤可略过）

```
apt-get autoremove
```

pascal 编译器和 ide，它们版本太老，干掉它们

```
rm -rf /usr/local/lib/lazarus
```

```
rm -rf /usr/local/lib/fpc
```

```
rm -rf /usr/lib/lazarus
```

```
rm -rf /usr/lib/fpc
```

## 修改内存交换文件(小内存系统，如 512M)

因为 IOT 的内存都是几百兆间，fpc 的编译过程很吃内存，没有大型交换文件系统，很容易造成崩溃+编译失败

如果你没有以 Root 方式登录，使用 vi 编辑

```
sudo vi /etc/dphys-swapfile
```

如果你以 root 方式登录，使用图形编辑器编辑，找到/etc/dphys-swapfile 来编辑即可

搜索 CONF\_SWAPSIZE 关键字，根据你的硬件配置，修改数值即可，建议数值 1000

```
CONF_SWAPSIZE=1000
```

然后保存并退出 vi

```
:wq
```

运行重启命令

```
sudo swapoff -a
```

```
sudo dphys-swapfile setup
```

```
sudo dphys-swapfile swapon
```

```
sudo swapon -a
```

## 安装附属小工具(此步骤可略过)

安装 git 工具，方便我们从 github 下载

```
apt-get install -y git-core
```

安装 svn 工具，方便我们从 sf.net 这类网站下载

```
apt-get install -y subversion
```

## 安装 gcc(此步骤可略过)

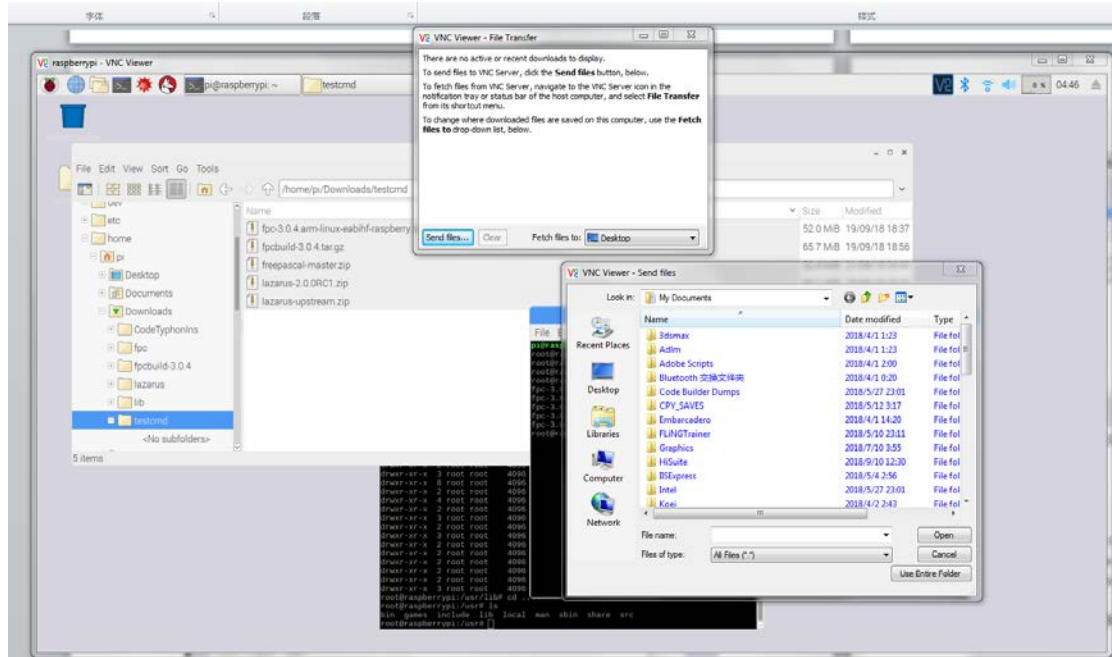
如果 ubuntu 的 linux 体系(debian,mate,ubuntu)安装 gcc 和，包括 make 那堆东西  
如果你的系统已经有 gcc 了，比如树莓派，香橙派，三星，可以跳过这一步

```
apt-get -y install build-essential
```

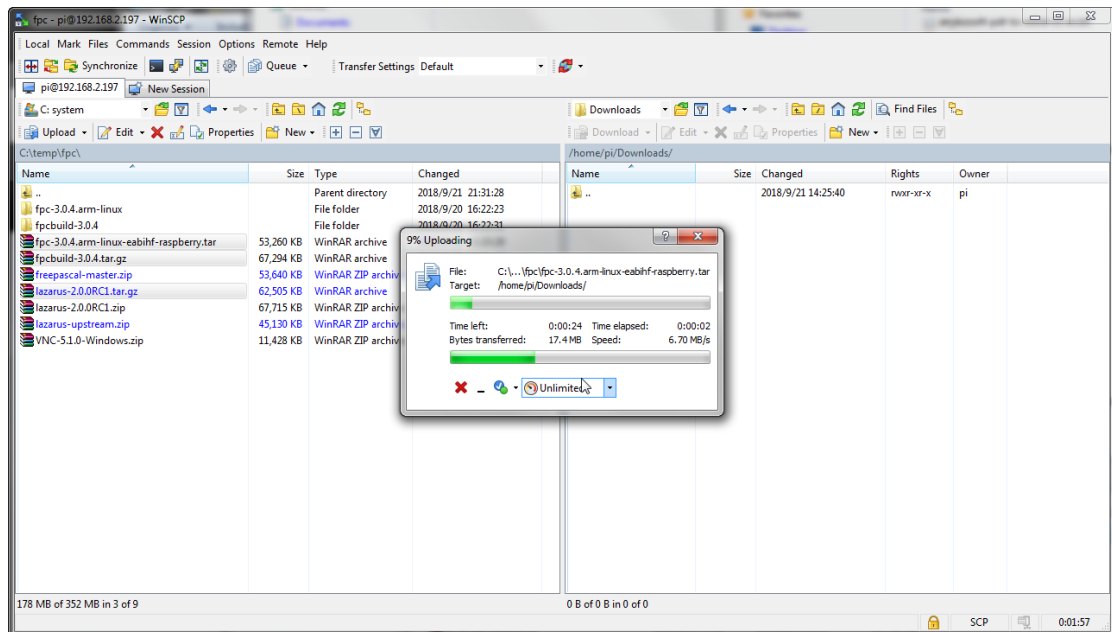


# 把 freepascal 和 lazarus 复制到 IOT 设备中

如果你有使用 vnc, 直接用 file transfer 即可, 如果是 ssh, 那么就用 filezilla,winscp 等工具, 传进来, 我们也可以 vnc 传输电脑的文件到树莓派系统



我们使用 winscp 在电脑直接访问 iot 的 ssh 服务器能做到同样的事



## 我们在树莓派的在文件管理器找到我们刚才传的 **fpc** 相关包

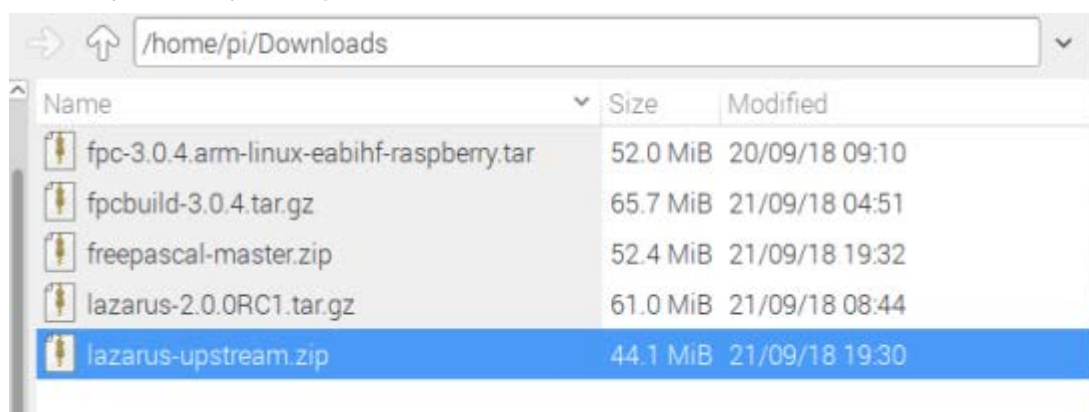
`fpc-3.0.4.arm-linux-eabi-hf-raspberry.tar` 这是针对树莓派 `arm linux` 定制的 `fpc` 编译器，自编译的编译器所需要的启动编译器，如果是编译 `github` 上同步推上来的最新版本，这个编译器只能在 `freepascal.org` 下载最新的，假如自编译器是 2.6.4 的，编译今天 `github` 刚下载的代码，是不会成功的，总之，`fpc` 的自编译器必须要求都是 `freepascal.org` 下载最新的

`fpcbuild-3.0.4.tar.gz` 这是我们刚下好的 `fpc` 编译器源代码

`lazarus-2.0.0RC1.tar.gz` 这是我们刚下好的 `lazarus` 的源代码

`freepascal-master.zip` 来自 `github` 的当日更新 `fpc` 源码

`lazarus-upstream.zip` 来自 `github` 的当日更新 `lazarus` 源码



在文件管理按 **F4** 打开控制台，

## 在 **root** 权限复制和解压

我们现在要在 `Linux` 创建一个跟文件夹，将我们的文件复制过去，并且解压

```
sudo su
```

```
mkdir /fpc
```

```
cp *.* /fpc
```

```
cd /fpc
```

`tar` 格式的压缩文件，以 `tar -xvf` 参数进行解压

```
tar -xvf fpc-3.0.4.arm-linux-eabi-hf-raspberry.tar
```

`gz` 格式的压缩文件，以 `tar -xzvf` 参数进行解压

```
tar -xzvf fpcbuild-3.0.4.tar.gz
```

```
tar -xzvf lazarus-2.0.0RC1.tar.gz
```

```
unzip freepascal-master.zip
```

```
unzip lazarus-upstream.zip
```

设备比较慢，从 `copy` 到解压，这一过程要持续好几分钟，待完成后，我们使用

```
ls -l
```

 查看我们在 `/fpc` 下的文件和目录，树莓派 `debian linux` 的 `ls -l` 等同于 `ll`

# 使用自编译方式：编译 fpc 编译器

如果你的 FPC 源码与自编译器一致，此步骤可以略过，直接去编译 LAZARUS 即可。如果需要编译 GITHUB 上最新的 FPC 编译器+LAZARUS 源码，以下方法可以在任何系统有效。

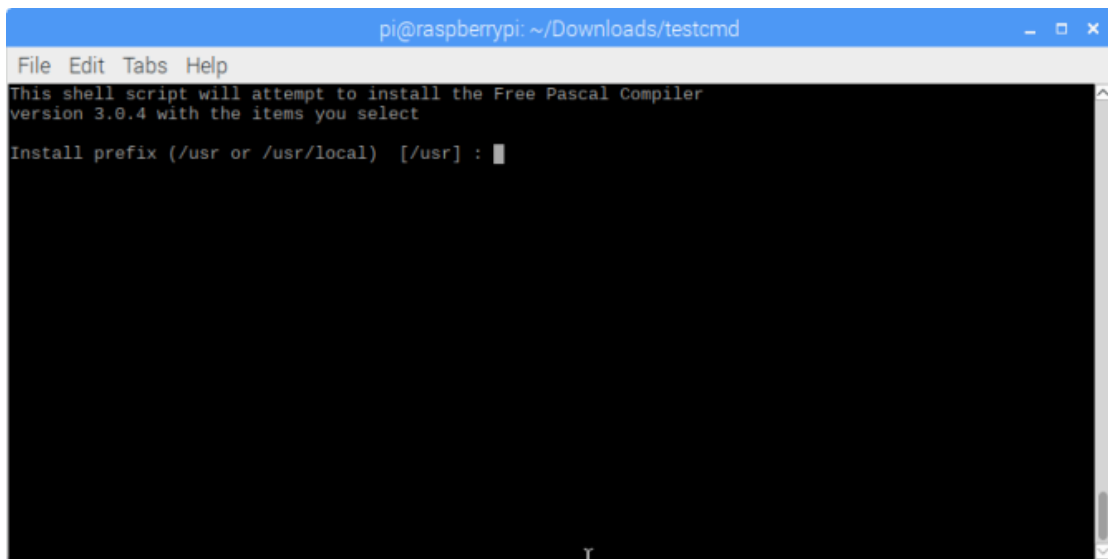
fpc-3.0.4.arm-linux 这个目录是我们的自编译所需要编译器，我们先安装它

```
sudo su
```

```
cd /fpc/fpc-3.0.4.arm-linux
```

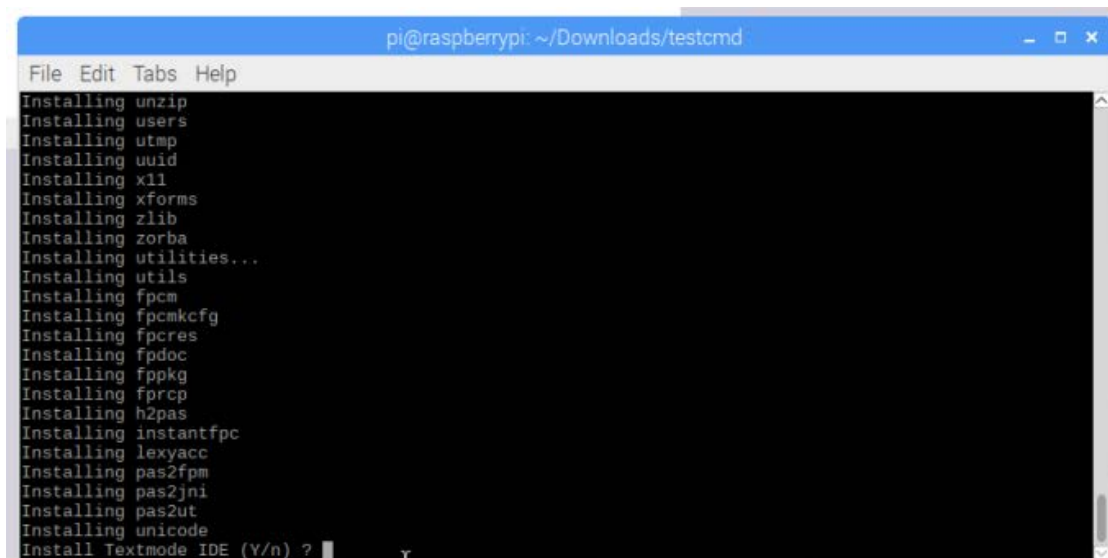
```
./install.sh
```

这时候，会出现路径前缀提示，直接回车安装过去（直接回车会安装到 root 权限的/usr 目录中，如果需要安装到当前用户的目录，就给/usr/local）



```
pi@raspberrypi: ~/Downloads/testcmd
File Edit Tabs Help
This shell script will attempt to install the Free Pascal Compiler
version 3.0.4 with the items you select
Install prefix (/usr or /usr/local) [/usr] : █
```

接下来，提示是否安装文本模式的 ide（类似 dos 时代的 turboc），我们选 y，安装它，因为等会我们会用它编写一个小型的测试程序来验证 fpc 编译器



```
pi@raspberrypi: ~/Downloads/testcmd
File Edit Tabs Help
Installing unzip
Installing users
Installing utmp
Installing uuid
Installing x11
Installing xforms
Installing zlib
Installing zorba
Installing utilities...
Installing utils
Installing fpcm
Installing fpcmcfg
Installing fpcres
Installing fpdoc
Installing fppkg
Installing fprcp
Installing h2pas
Installing instantfpc
Installing lexyacc
Installing pas2fpm
Installing pas2jni
Installing pas2ut
Installing unicode
Install Textmode IDE (Y/n) ? █
```

现在，它又提示我们是否安装文档，因为我们只用它中转过度来编译 fpc 的编译器，这里我们都开始选 n

```
pi@raspberrypi: ~/Downloads/testcmd
File Edit Tabs Help
Installing uuid
Installing x11
Installing xforms
Installing zlib
Installing zorba
Installing utilities...
Installing utils
Installing fpcm
Installing fpcmcfg
Installing fpcres
Installing fpdoc
Installing fppkg
Installing fprcp
Installing h2pas
Installing instantfpc
Installing lex yacc
Installing pas2fpm
Installing pas2jni
Installing pas2ut
Installing unicode
Install Textmode IDE (Y/n) ? n
Done.
Install documentation (Y/n) ?
```

安装 demo，因为我们只用它中转过度来编译 fpc 的编译器，这里我们都选 n

```
pi@raspberrypi: ~/Downloads/testcmd
File Edit Tabs Help
Installing xforms
Installing zlib
Installing zorba
Installing utilities...
Installing utils
Installing fpcm
Installing fpcmcfg
Installing fpcres
Installing fpdoc
Installing fppkg
Installing fprcp
Installing h2pas
Installing instantfpc
Installing lex yacc
Installing pas2fpm
Installing pas2jni
Installing pas2ut
Installing unicode
Install Textmode IDE (Y/n) ? n
Done.
Install documentation (Y/n) ? n
Install demos (Y/n) ?
```

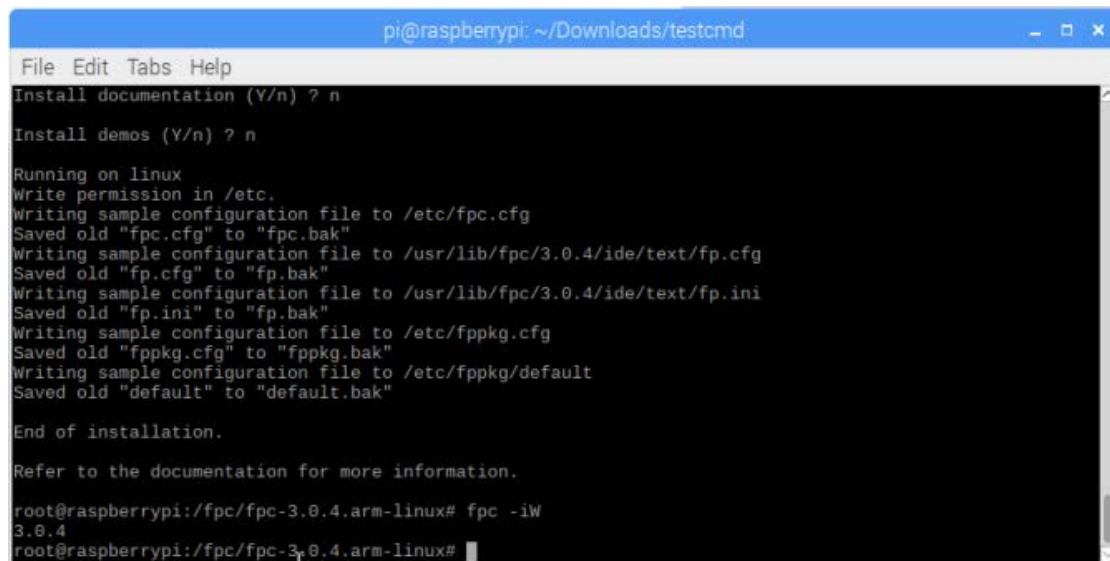
接下来，自编译器就安装成功了

```
pi@raspberrypi: ~/Downloads/testcmd
File Edit Tabs Help
Done.
Install documentation (Y/n) ? n
Install demos (Y/n) ? n
Running on linux
Write permission in /etc.
Writing sample configuration file to /etc/fpc.cfg
Saved old "fpc.cfg" to "fpc.bak"
Writing sample configuration file to /usr/lib/fpc/3.0.4/ide/text/fp.cfg
Saved old "fp.cfg" to "fp.bak"
Writing sample configuration file to /usr/lib/fpc/3.0.4/ide/text/fp.ini
Saved old "fp.ini" to "fp.bak"
Writing sample configuration file to /etc/fppkg.cfg
Saved old "fppkg.cfg" to "fppkg.bak"
Writing sample configuration file to /etc/fppkg/default
Saved old "default" to "default.bak"
End of installation.
Refer to the documentation for more information.
root@raspberrypi:/fpc/fpc-3.0.4.arm-linux#
```

好了，我们开始验证一下 fpc 的版本

`fpc -iW`

如果报告是 3.0.4 就说明是正确结果



```
pi@raspberrypi: ~/Downloads/testcmd
File Edit Tabs Help
Install documentation (Y/n) ? n
Install demos (Y/n) ? n
Running on linux
Write permission in /etc.
Writing sample configuration file to /etc/fpc.cfg
Saved old "fpc.cfg" to "fpc.bak"
Writing sample configuration file to /usr/lib/fpc/3.0.4/ide/text/fp.cfg
Saved old "fp.cfg" to "fp.bak"
Writing sample configuration file to /usr/lib/fpc/3.0.4/ide/text/fp.ini
Saved old "fp.ini" to "fp.bak"
Writing sample configuration file to /etc/fppkg.cfg
Saved old "fppkg.cfg" to "fppkg.bak"
Writing sample configuration file to /etc/fppkg/default
Saved old "default" to "default.bak"

End of installation.

Refer to the documentation for more information.

root@raspberrypi:/fpc/fpc-3.0.4.arm-linux# fpc -iW
3.0.4
root@raspberrypi:/fpc/fpc-3.0.4.arm-linux#
```

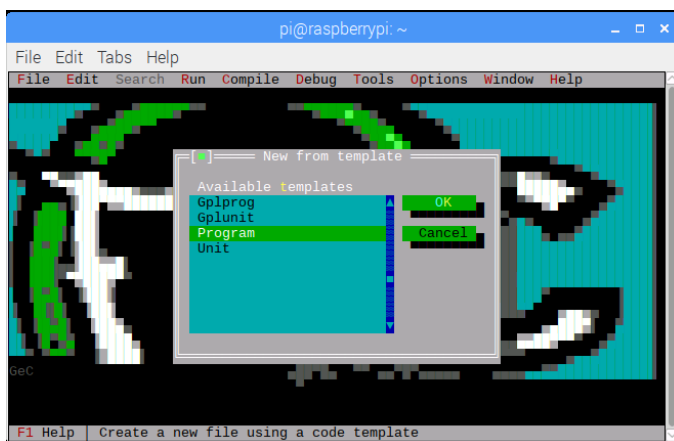
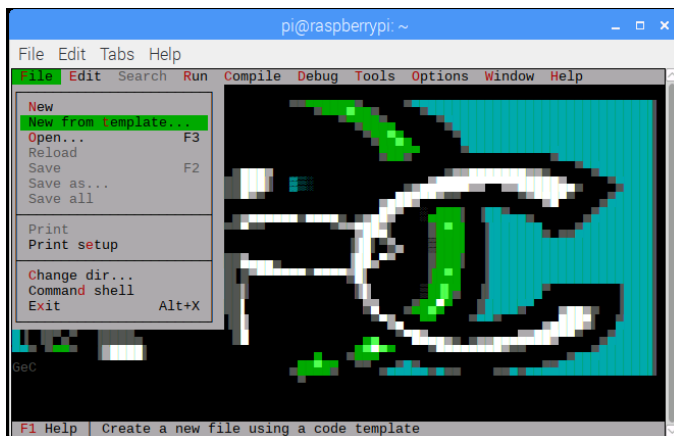
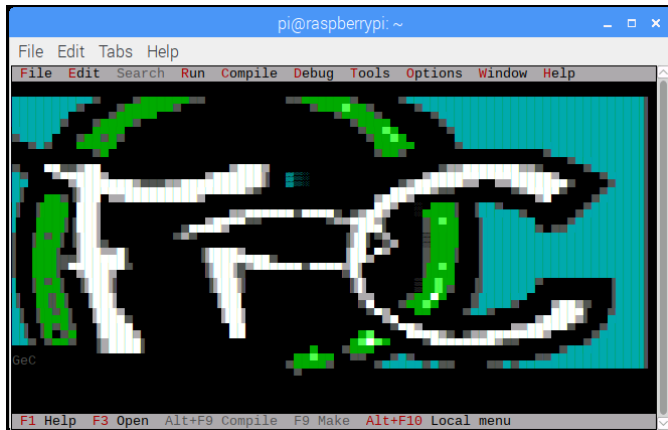
# 验证编译器是否支持自编译

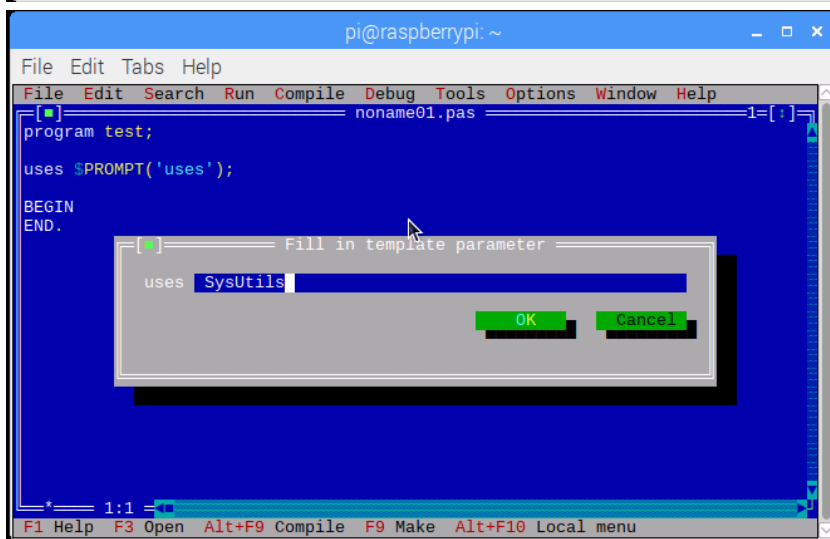
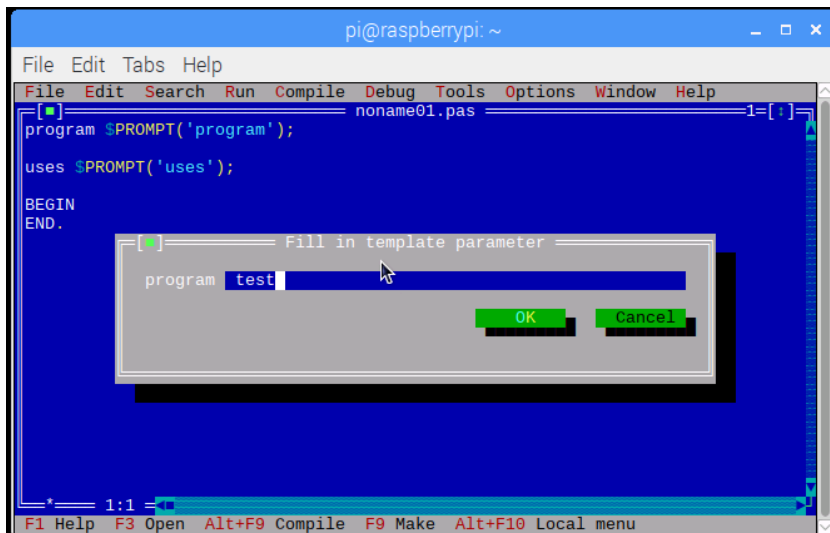
这是 diy 跨平台编译器和环境的良好习惯，避免后续出错节省编译时间  
打开 fpc 的文本模式 ide(注意：文本模式 ide 支持远程 TTY)

`sudo su`

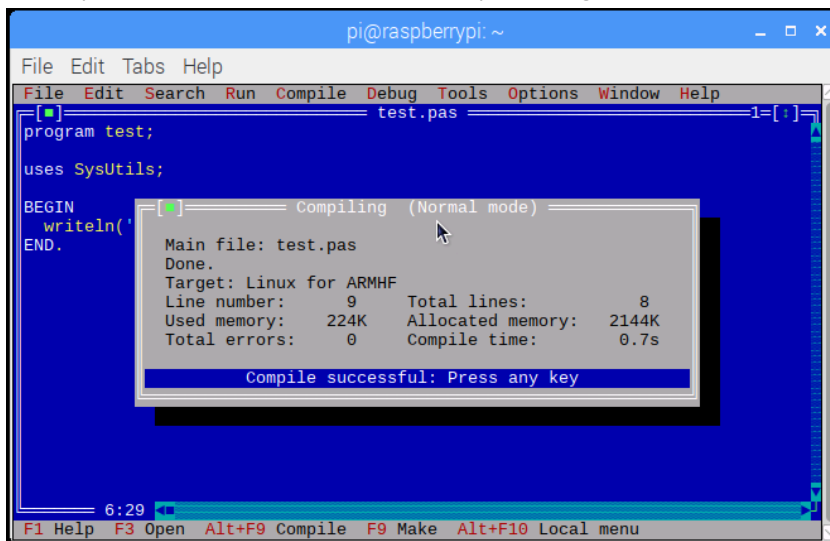
打开文本模式的 ide，如果 fp 无法被执行，就是 fpc 编译器和你系统以及 cpu 没对号。解决方案请访问 [freepascal.org](http://freepascal.org)

`fp`





如下图提示编译成功，那这个编译就是能用的。如果提示不成功，就是 fpc 编译器和你系统以及 cpu 没对号。解决方案请访问 [freepascal.org](http://freepascal.org)



验证编译以后，现在可以开始编译 fpc 源码了（此方法可用于 github 上的最新 fpc 源码）

# FPC 编译器所支持的 ARM 处理器 Model

下面罗列了 Fpc 编译器所支持的 arm 处理器架构标识符，给定了处理器标识符后，编译器生成的执行文件就会区分软件浮点和硬件浮点，cpu 指令内容一堆东西。经过证实，这些标识符架构都有效果，因为和这些 cpu 架构都有对应的 RT 运行库，所以最终的编译成败，要取决于实际编译结果。

- ARMV3
- ARMV4
- ARMV4T
- ARMV5
- ARMV5T
- ARMV5TEJ
- ARMV6: e.g. in Raspberry Pi; will probably work on most current hardware
- ARMV6K
- ARMV6T2
- ARMV6Z
- ARMV7: e.g. in Nokia N900, Nokia N9, Nokia N950
- ARMV7A: e.g. in Odroid U2, Odroid U3
- ARMV7R
- ARMV7M
- ARMV7EM



## 编译 fpc(编译大规模部署版本 3.0.4)

`sudo su`

`cd /fpc/fpcbuild-3.0.4/fpcsrc`

如果 arm 处理器是非 armv7 我们只能使用 armv6 构建

`make install sourceinstall OPT="-dFPC_ARMHF -CpARMV6 -OpARMV6" PREFIX=/usr`

如果 arm 处理器是 armv7

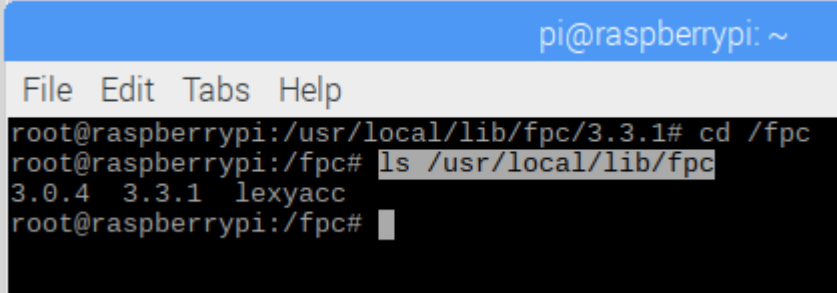
`make install sourceinstall OPT="-dFPC_ARMHF" PREFIX=/usr`

这一过程持续时间很长，大概 20 分钟以上，上个厕所，喝杯茶，随便干点什么事回来

20 分钟以后，编译完成

我们开始罗列已有的 fpc 编译器

`ls /usr/lib/fpc`



```
pi@raspberrypi: ~  
File Edit Tabs Help  
root@raspberrypi:/usr/local/lib/fpc/3.3.1# cd /fpc  
root@raspberrypi:/fpc# ls /usr/local/lib/fpc  
3.0.4 3.3.1 lexyacc  
root@raspberrypi:/fpc#
```

现在，我们定位的新编译器是 3.0.4

先删除系统中老的 fpc 编译器

`rm -f /usr/bin/ppcarm`

替代我们刚编译好的编译器

`cp /usr/lib/fpc/3.0.4/ppcarm /usr/bin/ppcarm`

最后查看 fpc 编译器的版本以及支持架构和操作系统

`fpc -i`

重新生成 fpc 的编译核心参数配置

`fpcmkcfg -d basepath=/usr/lib/fpc/3.0.4/ > /etc/fpc.cfg`

fpc3.0.4 的编译器构建成功

## 编译 fpc(来自 github 的当天更新版本)

```
sudo su
cd /fpc/freepascal-master
```

如果 arm 处理器是非 armv7 我们只能使用 armv6 构建

```
make install sourceinstall OPT="-dFPC_ARMHF -CpARMV6 -OpARMV6" PREFIX=/usr
```

如果 arm 处理器是 armv7

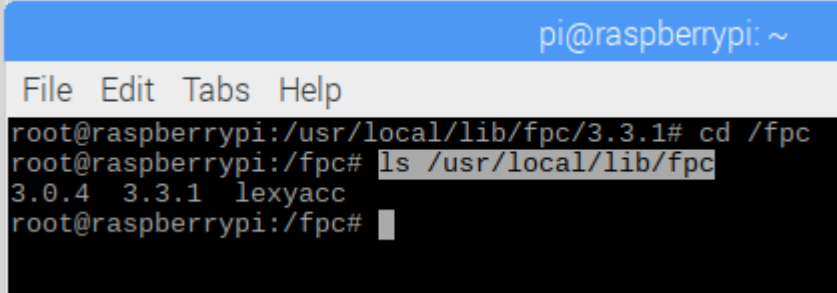
```
make install sourceinstall OPT="-dFPC_ARMHF" PREFIX=/usr
```

这一过程持续时间很长, 大概 20 分钟以上, 上个厕所, 喝杯茶, 随便干点什么事回来

20 分钟以后, 编译完成

我们开始罗列已有的 fpc 编译器

```
ls /usr/lib/fpc
```



```
pi@raspberrypi: ~
File Edit Tabs Help
root@raspberrypi:/usr/local/lib/fpc/3.3.1# cd /fpc
root@raspberrypi:/fpc# ls /usr/local/lib/fpc
3.0.4 3.3.1 lexyacc
root@raspberrypi:/fpc#
```

现在, 我们定位的新编译器是 3.3.1

先删除系统中老的 fpc 编译器

```
rm -f /usr/bin/ppcarm
```

替代我们刚编译好的编译器

```
cp /usr/lib/fpc/3.3.1/ppcarm /usr/bin/ppcarm
```

最后查看 fpc 编译器的版本以及支持架构和操作系统

```
fpc -i
```

重新生成 fpc 的编译核心参数配置

```
fpcmkcfg -d basepath=/usr/lib/fpc/3.3.1/ > /etc/fpc.cfg
```

来自 github 的最新 fpc 编译器构建成功

# 编译和安装 Lazarus 的 IDE

本章节成功编译 Lazarus 所使用的版本来自 github 下载  
<https://github.com/graemeg/lazarus>

本章节成功编译 Lazarus2.1.0 所使用的 fpc 版本为 3.3.1

如果已经是 root 模式了，下面这句可以省略

```
sudo su
```

定位 lazarus 当前目录

```
cd /fpc/lazarus-upstream
```

开始编译 lazarus

如果 arm 处理器是非 armv7 我们只能使用 armv6 构建

```
make OPT="-dFPC_ARMHF -CpARMV6 -OpARMV6"
```

如果 arm 处理器是 armv7

```
make OPT="-dFPC_ARMHF"
```

这一过程大概 20 分钟左右

完成以后启动 lazarus

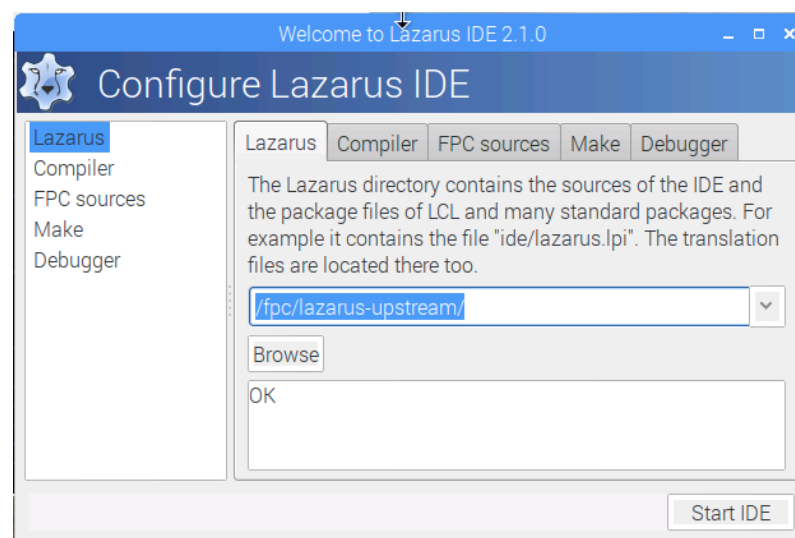
```
sudo su
```

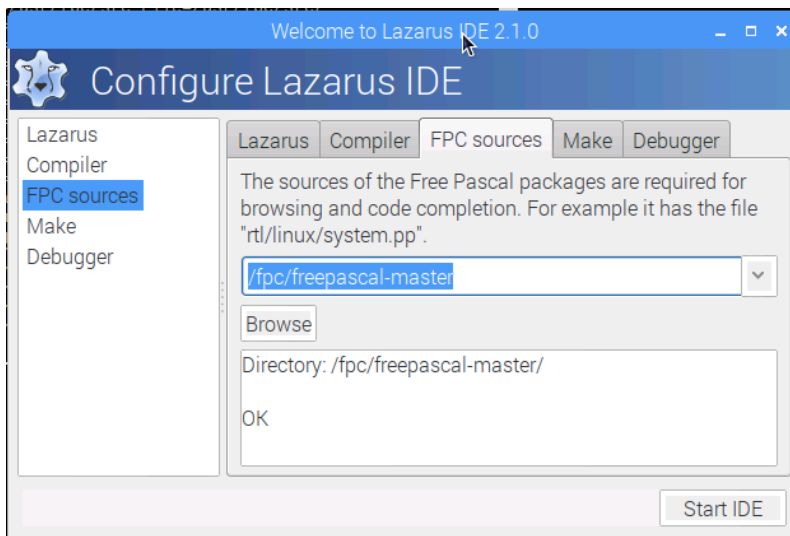
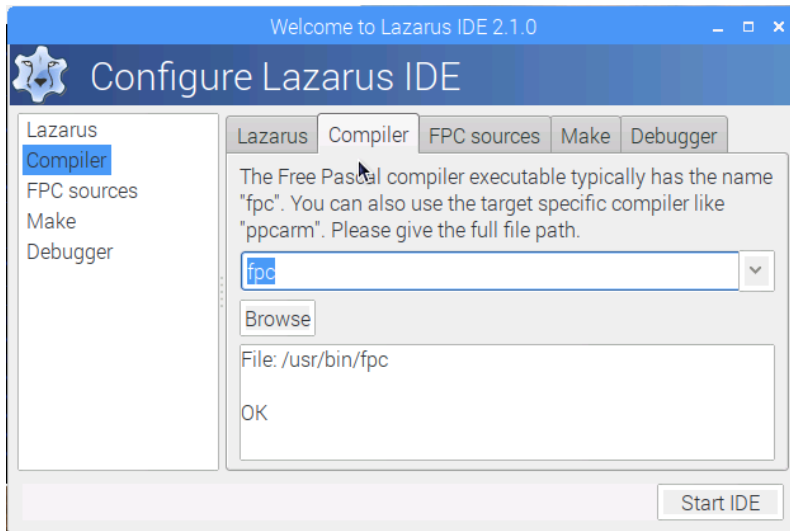
```
cd /fpc/lazarus-upstream
```

```
./lazarus
```

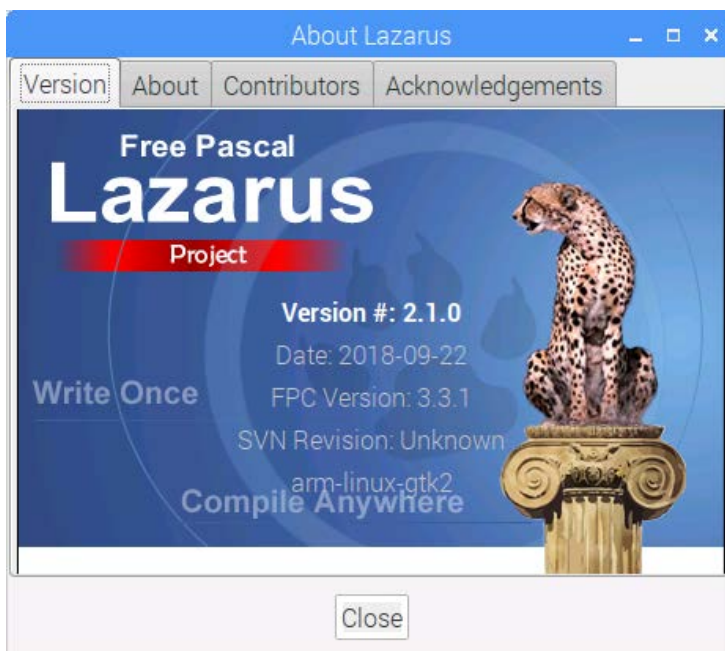
启动界面会显示 lazarus 的版本为 2.1.0

然后提示输入路径



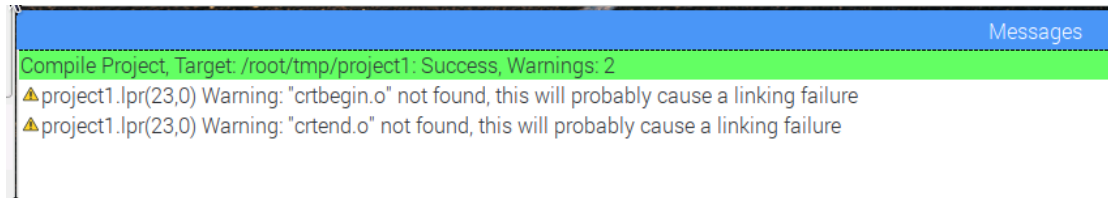


正确填写对应的路径，然后 **Start IDE**，进去以后，就是 **lazarus2.1.0** 了  
打开 **help->about** 对话框，显示 **fpc** 版本以及 **lazarus** 版本，这些都是我们刚才编译的东西



# fpc 编译时报错：找不到 crtbegin.o

当我们编译程序时，FPC 编译器提示找不到 `CRTBEGIN.O+CRTEND.O`

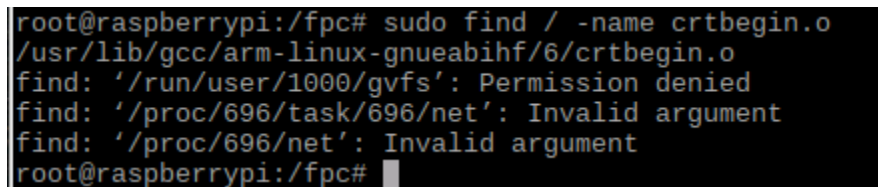


```
Messages
Compile Project, Target: /root/tmp/project1: Success, Warnings: 2
▲project1.lpr(23,0) Warning: "crtbegin.o" not found, this will probably cause a linking failure
▲project1.lpr(23,0) Warning: "crtend.o" not found, this will probably cause a linking failure
```

解决办法

`sudo su`

`find / -name crtbegin.o`



```
root@raspberrypi:/fpc# sudo find / -name crtbegin.o
/usr/lib/gcc/arm-linux-gnueabi/6/crtbegin.o
find: '/run/user/1000/gvfs': Permission denied
find: '/proc/696/task/696/net': Invalid argument
find: '/proc/696/net': Invalid argument
root@raspberrypi:/fpc#
```

Find 给出反馈，那个 `crtbegin.o` 文件的位置，我们路径粘贴下来

`/usr/lib/gcc/arm-linux-gnueabi/6/`

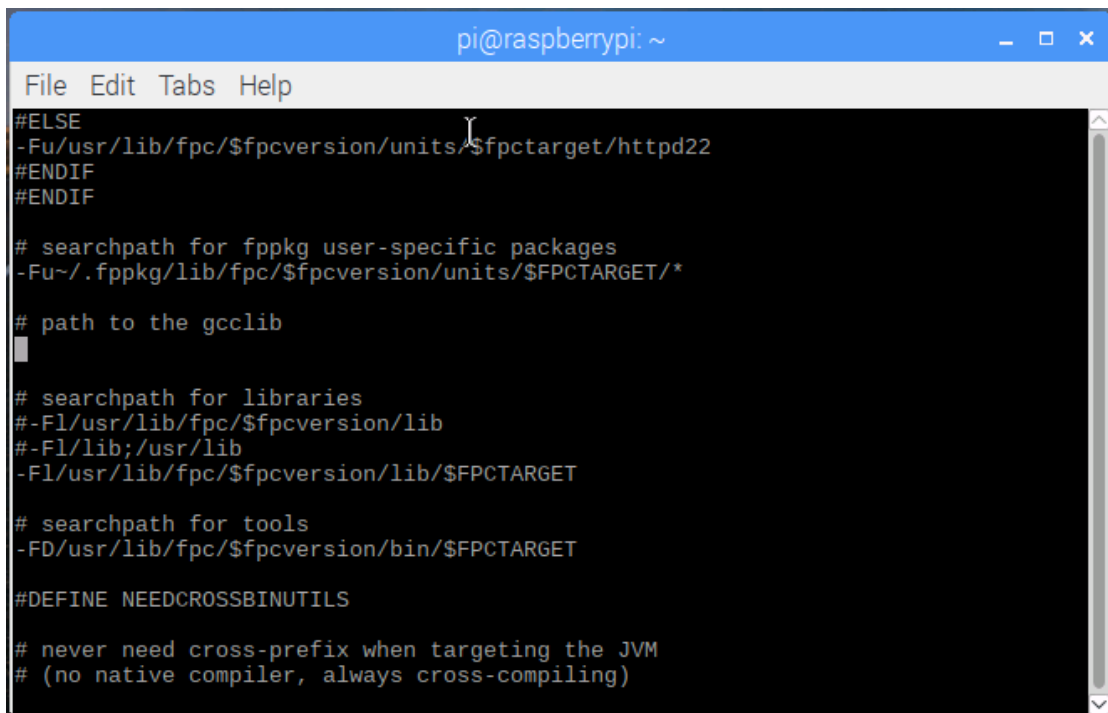
然后，我们打开 fpc 的编译器配置文件

`sudo su`

`vi /etc/fpc.cfg`

在 vi 中，查找关键字 `# path to the gcclib`

提示：如果使用 `root` 身份，可以直接通过图形编辑器来配置，操作比 `vi` 简单许多



```
pi@raspberrypi: ~
File Edit Tabs Help
#ELSE
-Fu/usr/lib/fpc/$fpcversion/units/$fpc$target/httpd22
#ENDIF
#ENDIF

# searchpath for fppkg user-specific packages
-Fu~/fppkg/lib/fpc/$fpcversion/units/$FPCTARGET/*

# path to the gcclib
█

# searchpath for libraries
#-Fl/usr/lib/fpc/$fpcversion/lib
#-Fl/lib;/usr/lib
-Fl/usr/lib/fpc/$fpcversion/lib/$FPCTARGET

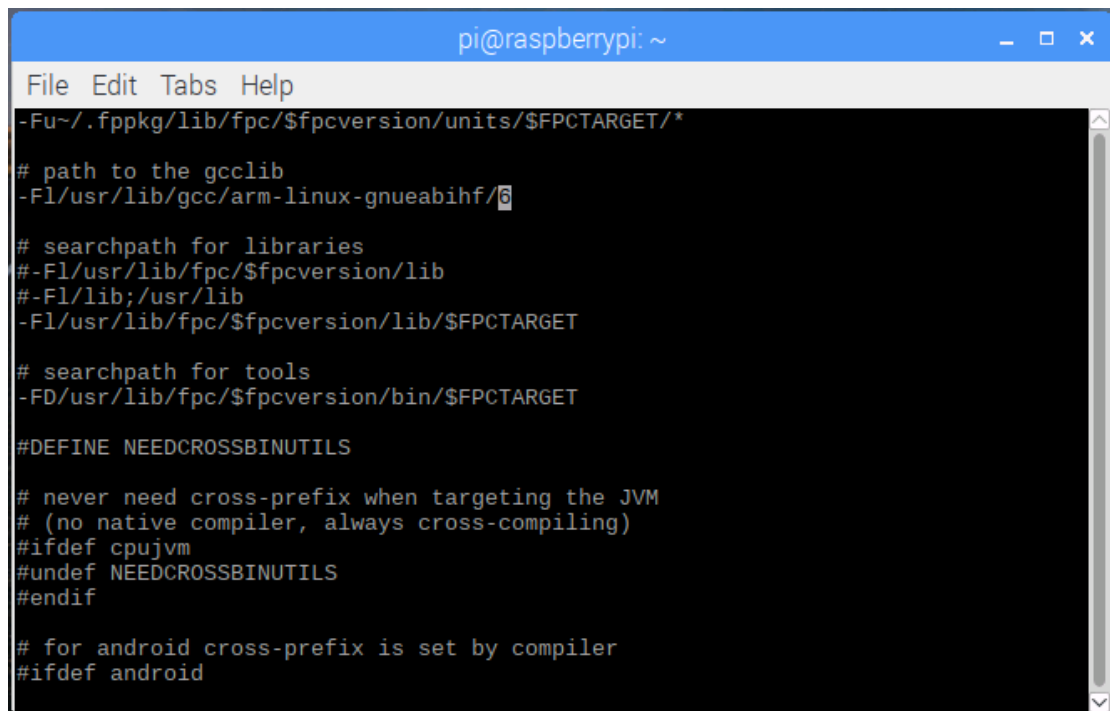
# searchpath for tools
-FD/usr/lib/fpc/$fpcversion/bin/$FPCTARGET

#DEFINE NEEDCROSSBINUTILS

# never need cross-prefix when targeting the JVM
# (no native compiler, always cross-compiling)
```

按下 i 键，进入插入模式

`-Fl/usr/lib/gcc/arm-linux-gnueabi/hf/6`



```
pi@raspberrypi: ~
File Edit Tabs Help
-Fu~/fppkg/lib/fpc/$fpcversion/units/$FPCTARGET/*
# path to the gcclib
-Fl/usr/lib/gcc/arm-linux-gnueabi/hf/6
# searchpath for libraries
#-Fl/usr/lib/fpc/$fpcversion/lib
#-Fl/lib;/usr/lib
-Fl/usr/lib/fpc/$fpcversion/lib/$FPCTARGET
# searchpath for tools
-FD/usr/lib/fpc/$fpcversion/bin/$FPCTARGET
#DEFINE NEEDCROSSBINUTILS
# never need cross-prefix when targeting the JVM
# (no native compiler, always cross-compiling)
#ifdef cpujvm
#undef NEEDCROSSBINUTILS
#endif
# for android cross-prefix is set by compiler
#ifdef android
```

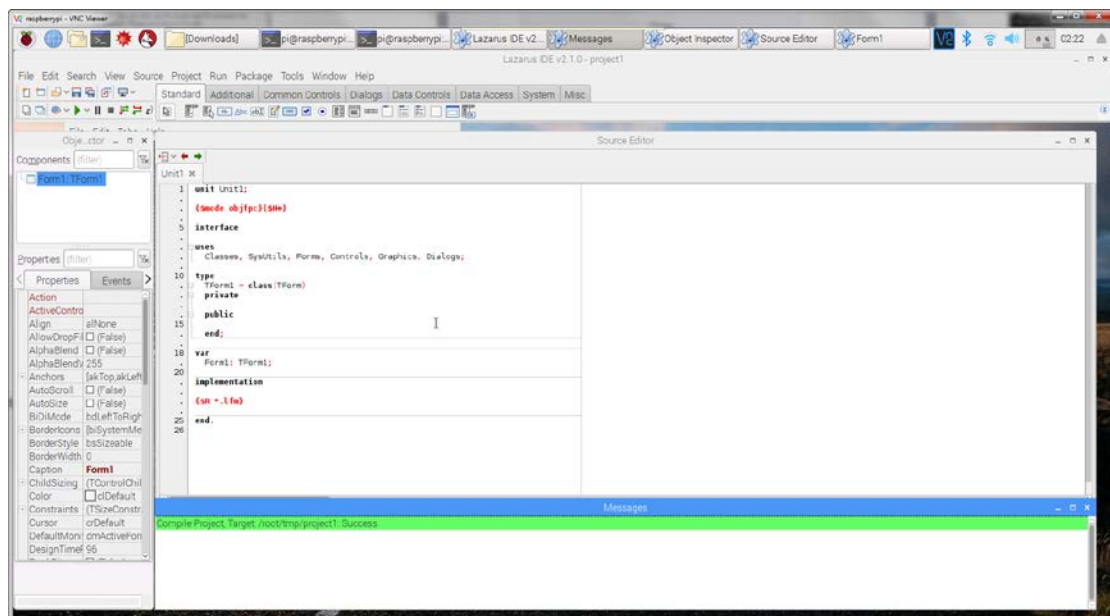
按下 esc 键退出插入模式

再用下列命令保存 fpc.cfg 退出 vi 编辑器

`:wq`

然后重开 lazarus，重新编译我们的工程

提示编译成功，fpc 编译器内核版本就是刚才 diy 出来的 3.3.1



至此，所有 lazarus 以及 fpc 编译器的构建就全部完成了

# Freepascal 的 IOT 通讯组件

IOT 平台的 pascal 通讯引擎可访问我们的开源地址

<https://github.com/PassByYou888/ZServer4D>

## 关于 fpc+lazarus 构建中文指南作者

我是原创，有事可来给我留言

600585@qq.com

By600585

2018-9-22