二代龙芯派上创建 Linux From Scratch 操作实录

zevan(QQ: 85948681)

前言

本文记录了在二代龙芯派上创建 Linux From Scratch（LFS）的全过程，以及在应用过程中遇到的一些问题。不保证对其他用户一定适用，仅供参考。

1. 准备工作

全新的龙芯派存在一些问题，诸如每次开机随机生成一个网卡的 MAC 地址，导致 DHCP 服务器每次都会给龙芯派分配新的 IP，使远程登录变得不方便；而且，龙芯派自带的 SSD 硬盘只有 16GB，空间太小，要构建 LFS 得换更大的 SSD 硬盘。

1.1 **更新 pmon（可选操作）**

小白用户请跳过这一步操作。

警告：高危操作，有变砖风险！

这里，使用了 flygoat 提供的 2.3 版本的 pmon。

https://github.com/FlyGoat/pmon-ls2k-pi2/releases/tag/v2.3

放在 U 盘里，确保 md5 值准确无误，然后用 load 命令刷。强烈建议对文件进行 md5 校验，确保 pmon 固件下载、复制都正确。

建议用最皮实可靠的 U 盘进行操作，一定不要用不稳定的 U 盘来操作！

load -r -f 0xbfc00000 (usb0,0)/flash.bin

1.2 设置 MAC 地址

用 setmac 设置一下网卡的 MAC 地址，避免每次开机就更换一个 IP

PMON> setmac syn1 "f2:24:ee:e4:47:53"
set syn1 Mac address: f2:24:ee:e4:47:53
The machine should be restarted to make the mac change to take effect!!
PMON> setmac syn0 "f1:34:ee:e5:57:56"
set syn0 Mac address: f1:34:ee:e5:57:56
The machine should be restarted to make the mac change to take effect!!

http://www.loongnix.org/index.php?title=Pmon%E5%B8%B8%E7%94%A8%E5%91%BD%E4%BB%A4&oldid=3456

由于 pmon 中有 bug，实际网卡的物理地址并不是此时设置的地址。不过，地址总算是可以固定下来了。

1.3 **安装 Loognix 操作系统**

我买了一块 120GB 的 SSD 硬盘，替换了二代龙芯派自带的 16GB SSD 硬盘。接下来需要给新 SSD 硬盘安装系统。

首先，下载 Loognix 20180930 版本的 iso 文件。

http://ftp.loongnix.org/os/loongnix/1.0/liveinst/old/loongnix-20180930.iso

记得对 ISO 文件进行 MD5 校验

http://ftp.loongnix.org/os/loongnix/1.0/liveinst/old/loongnix-20180930.iso.md5

用 dd 命令制作安装 U 盘。安装细节不再赘述，详情可以参考鸽工的视频教程

龙芯 TechLive 第一期：安装 Loongnix 系统 https://www.bilibili.com/video/av57802116

安装的时候在分区的时候需要小心一点：

/boot 分布必须是第一个分区。分区大小 500MB 左右即可，一定不能用 ext4 格式，建议用 ext3 格式。

/ 分区

/home 分区，可以让多个系统共享一个 home

分别给 debian 和 LFS 的安装留出分区。我给 debian 和 LFS 都预留了 20GB 的分区，挂载在 /DEBIAN 和/LFS 上。

swap 分区留了 2GB，和内存大小一致。

安装过程中，遇到了两个问题：

第一个问题：龙芯派比较挑 HDMI 线，使用绿联的 HDMI 线可以正常显示，使用渣渣显示器自带的线不能显示画面。如果发现龙芯派屏幕不亮，很可能是 HDMI 线的问题，而不是派的问题。

第二个问题：两个标准 USB 口，如果同时插上 USB 无线鼠标和 USB 键盘，pmon 会报错，无法正常启动。建议只插一个 USB 键盘，等正常载入系统的时候再插鼠标等外设。

花了半个小时左右，安装成功。可以正常启动。

配置 SSH

systemctl enable sshd.service

systemctl start sshd.service

默认开机进入命令行界面

systemctl set-default multi-user.target

升级一些必要的软件

yum install gcc

然而我并没有使用 Loongnix 来做 LFS，我更喜欢 debian，所以我又装了一个 Debian 系统。Loongnix 内有 GCC 7.2，版本更高，可能更适合做 LFS。在 Loongnix 系统下，理论上也可以用类似的流程构建 LFS。

1.5 安装 Debian

在 Loognix 环境中，用 debootstrap 方法安装 debian buster。注意，在准备安装 debian 的分区内进行操作。

yum install -y debootstrap

以安装 debootstrap

debootstrap --arch mips64el buster http://ftp.cn.debian.org/debian/

1.7 更新一下内核

Loongnix 自带的内核有一个问题，开机的时候必须连上显示器，且打开显示器，否则不能开机。这个问题理论上可以自行编译新版本的内核来解决。我采用了一个更简单的做法，做伸手党，请 flygoat 编译了一个龙芯派的内核，版本 4.19。下载地址如下：

https://repo.flygoat.com/pub/~jiaxun/vmlinuz-2k-dvo1-1080p-wireless

由于版权问题，该内核不完美，无法正常关机，需要长按电源键关机，请酌情使用。

1.6 配置 boot.cfg

配置 Debian 使用 Loongnix 的内核来启动系统，串口和屏幕同时输出。注意，Loognix 自带的内核中存在一点 bug，启动的时候必须打开显示器，否则无法进入系统。这个 bug 可以通过自行编译内核解决，请参考本文中内核编译部分。

```
timeout 5
default 0
showmenu 1

title 'Loongnix GNU/Linux'
	kernel (wd0,0)/vmlinuz-3.10.0-1.fc21.loongson.2k.11.mips64el
	initrd (wd0,0)/initramfs-3.10.0-1.fc21.loongson.2k.11.mips64el.img
	args	root=/dev/sda6 ro rhgb quiet loglevel=0 LANG=zh_CN.UTF-8 console=ttyS0,115200 console=tty

title 'Loongnix GNU/Linux rescue'
	kernel (wd0,0)/vmlinuz-0-rescue-9d0d90497b314a4d96c7154666c951ec.2k
	initrd (wd0,0)/initramfs-0-rescue-9d0d90497b314a4d96c7154666c951ec.2k.img
	args	root=/dev/sda6 ro rhgb quiet loglevel=0 LANG=zh_CN.UTF-8 console=ttyS0,115200 console=tty

title 'Debian GNU/Linux (10.0) 3.10.0-1'
	kernel (wd0,0)/vmlinuz-3.10.0-1.fc21.loongson.2k.11.mips64el
	initrd (wd0,0)/initramfs-3.10.0-1.fc21.loongson.2k.11.mips64el.img
	args	root=/dev/sda3 ro rhgb quiet loglevel=0 LANG=zh_CN.UTF-8 console=ttyS0,115200 console=tty

title 'Debian GNU/Linux (10.0) rescue'
	kernel (wd0,0)/vmlinuz-0-rescue-9d0d90497b314a4d96c7154666c951ec.2k
	initrd (wd0,0)/initramfs-0-rescue-9d0d90497b314a4d96c7154666c951ec.2k.img
	args	root=/dev/sda3 ro rhgb quiet loglevel=0 LANG=zh_CN.UTF-8 console=ttyS0,115200 console=tty
```

如果要在 pmon 中手动引导内核启动，需要使用的命令如下：

```
load (wd0,0)/boot/vmlinuz-3.10.0-1
Initrd (wd0,0)/boot/initramfs-3.10.0-1
g console=ttyS0,115200 root=/dev/sda6 ro rhgb quiet loglevel=0 LANG=zh_CN.UTF-8
```

1.7 **修改 fstab**

为了增强系统的健壮性，避免因为断电而导致出现无法开机的情况，修改 fstab 文件

https://help.ubuntu.com/community/Fstab

https://os.mbed.com/handbook/Mounting-with-sync

https://unix.stackexchange.com/questions/146620/difference-between-sync-and-async-mount-options

加上 errors=remount-ro 参数，为了更安全可以加上 sync 选项，代价是系统响应速度可能更慢，SSD 硬盘的寿命更低。

```
UUID=30fcb748-ad1e-4228-af2f-951e8e7b56df / ext3 defaults,errors=remount-ro,noatime，sync 0 1
```

1.8 **准备内核源码（可选）**

从 loongnix 的 git 库里下载内核源码，分支选 release-1903

http://cgit.loongnix.org/cgit/linux-3.10/

```
git checkout -b release-1903
```

不过，我并没有自己编译内核，而是使用了上面提到的 4.19 内核。

2. 编译 LFS

2.1 资料收集

编译 LFS 可以参考这两位前辈的经验

https://blog.csdn.net/Lina_ACM/article/details/79736930

https://blog.csdn.net/Lina_ACM/article/details/79670603

https://www.yhi.moe/cn/2018/12/23/lfs-on-mips64-process.html

国内社区翻译的中文版的 LFS 9.0

https://lctt.github.io/LFS-BOOK/lfs-sysv/index.html

以及航天龙梦孙海勇的《手把手教你构建自己的 LINUX 系统》

2.2 环境设置以及文件下载

把给 LFS 准备的分区挂载到/mnt/lfs。设置 LFS 环境变量为/mnt/lfs

```
export LFS=/mnt/lfs
```

参考了前辈的经验，我对 LFS_TGT 变量做了一点点修改，改为

```
LFS_TGT=mips64el-lfs-linux-gnu
```

创建$LFS/sources 文件夹

下载所有 LFS 需要的文件

```
wget --input-file=wget-list --continue --directory-prefix=$LFS/sources
```

有一些文件不能正常下载，需要换一个镜像来下载，比如在
http://www.linuxfromscratch.org/mirrors.html 中的 files mirrors。
建议访问国内镜像
https://mirrors.ustc.edu.cn/lfs/lfs-packages/9.0/
https://mirrors.ustc.edu.cn/lfs/lfs-packages/lfs-packages-9.0.tar

2.3 构建临时环境

2.3.1 编译 binutils pass1

编译耗时 18 分钟（只看 make 的耗时），注意编译的时候加上-j3 的参数，使 make 可以并行进行，能够节约一半的编译时间。

据此估算，如果加上 configure 的时间，编译 binutils 的耗时，即一个 SBU，大约是 19 分钟。根据 SBU 的时间，可以估计其他程序编译的耗时。

我收集了一下各种 CPU 编译 LFS 时一个 SBU 的大小，以此可以估算龙芯 2K 1000 的性能大概相当于什么处理器。

表 1 不同型号的处理器 SBU 时间

处理器型号	主频（GHz）	核数	SBU
龙芯 2K1000（龙芯派）	1.0	2	19m
BCM2711 SoC （树莓派 4）	1.5	4	4m
AMD Athlon X2	2.5	2	15m
Intel Celeron N2840	2.16	2	6m17.613s
AMD FX-8350 (over clock)	4.4	8	1.23m
Intel Atom 330	1.6	2	19m
BCM2835（Raspberry Pi Zero W）	0.7	1	58m51.823s
Intel Xeon E5-1650	3.2	6	2m37s(without -j)
PowerPC G5	2.0	2	4m17.748s
Intel Core i5-3470	3.2	4	0m41.302s
Intel Core i3-550	3.2	2	1m38.310s
Qualcomm Snapdragon 410 MSM8916	1.2	4	30m51.855s
Intel Core i5-6300HQ	2.3	4	0m42.588s

数据来源：
https://www.linuxquestions.org/questions/linux-from-scratch-13/what-cpu-are-you-using-and-what-is-your-sbu-4175634812/page2.html

2.3.2 GCC pass 1

根据 LFS 手册，GCC pass 1 构建需要 12 SBU，大概需要耗时 228 分钟。

在编译之前，需要作出修改。在 x86_64 下编译 LFS，需要修改三个文件
gcc/config/linux.h,
gcc/config/i386/linux.h, 和 gcc/config/i368/linux64.h。

对于 MIPS 下的编译，需要修改的是 gcc/config/linux.h 和 gcc/config/mips/linux.h 两个文件,相应的脚本为：

```
for file in gcc/config/linux.h gcc/config/mips/linux.h
do
    cp -uv $file{,.orig}
    sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
    -e 's@/usr@/tools@g' $file.orig > $file
    echo '
    #undef STANDARD_STARTFILE_PREFIX_1
    #undef STANDARD_STARTFILE_PREFIX_2
    #define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
    #define STANDARD_STARTFILE_PREFIX_2 "" >> $file
    touch $file.orig
done
```

编译的时候，需要根据龙芯的环境作出修改，参考 Yhi Junde 的 blog、
https://www.yhi.moe/cn/2018/12/23/lfs-on-mips64-process.html
修改如下：

```
../configure \
    --target=$LFS_TGT \
    --prefix=/tools \
    --with-glibc-version=2.11 \
    --with-sysroot=$LFS \
    --with-newlib \
    --without-headers \
    --with-local-prefix=/tools \
    --with-native-system-header-dir=/tools/include \
    --disable-nls \
    --disable-shared \
    --disable-multilib \
    --disable-decimal-float \
    --disable-threads \
    --disable-libatomic \
    --disable-libgomp \
    --disable-libquadmath \
    --disable-libssp \
    --disable-libvtv \
    --disable-libstdcxx \
    --enable-languages=c,c++ \
    --with-abi=64 --with-arch=mips64r2 --with-tune=loongson3a
```

针对龙芯 2K，可能有特殊的 --with-arch 和--with-tune 参数。为了保险起见，先不加太多优化。

另外，并没有增加 --build 选项

预计耗时 228 分钟，实际耗时 203 分钟。

后续的编译速度都比较快，最慢的就是 GCC 的编译。

2.3.3 Linux-3.10 API headers

这里没有用 Linux 5.2 的内核源码，用的是龙芯的 3.10 内核源码。

git clone 下 release-1903 版本的 Linux 3.10 内核源码，然后按照 LFS 手册来操作。

2.3.4 Bison

Make check 的时候出错。

```
make[3]: 'examples/c/calc/calc' is up to date.
  LEX          examples/c/lexcalc/scan.c
  CC           examples/c/lexcalc/lexcalc-scan.o
gcc: error: ./examples/c/lexcalc/scan.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
```

暂时不知道是否有影响。

2.3.5 Python

编译的时候出了点问题，不过还是结束了。

```
/mnt/lfs/sources/Python-3.7.4/Modules/nismodule.c:17:10: fatal error: rpc/rpc.h: No such file or
directory
    17 | #include <rpc/rpc.h>
       |          ^~~~~~~~~~
compilation terminated.
INFO: Could not locate ffi libs and/or headers
```

Make install 的时候报错

```
Traceback (most recent call last):
  File "/mnt/lfs/sources/Python-3.7.4/Lib/runpy.py", line 193, in _run_module_as_main
    "__main__", mod_spec)
  File "/mnt/lfs/sources/Python-3.7.4/Lib/runpy.py", line 85, in _run_code
    exec(code, run_globals)
  File "/mnt/lfs/sources/Python-3.7.4/Lib/ensurepip/__main__.py", line 5, in <module>
    sys.exit(ensurepip._main())
  File "/mnt/lfs/sources/Python-3.7.4/Lib/ensurepip/__init__.py", line 204, in _main
    default_pip=args.default_pip,
  File "/mnt/lfs/sources/Python-3.7.4/Lib/ensurepip/__init__.py", line 117, in _bootstrap
    return _run_pip(args + [p[0] for p in _PROJECTS], additional_paths)
  File "/mnt/lfs/sources/Python-3.7.4/Lib/ensurepip/__init__.py", line 27, in _run_pip
    import pip._internal
zipimport.ZipImportError: can't decompress data; zlib not available
make: *** [Makefile:1132: install] Error 1
```

不管了，好像不影响，可以继续。

2.3.6 Stripping

跳过，不做了。

至此，LFS 的临时系统已经全部完成了。每个软件包编译的耗时间在表 2 中列出，耗时超过 1 SBU 的软件包已经标红。可见，大部分的程序编译很块，最慢的是 GCC 的编译。

表 2 临时系统软件包构建耗时统计

软件包	make 耗时
binutils pass 1	18m
gcc 9.2 pass 1	203m
glibc-2.30	66m
libstdc++	6m56.376s
Binutils pass 2	18m44.005s
gcc 9.2 pass 2	263m18.820s
tcl	22m35.361s
expect	0m37.312s
Deja GNU	0m0.047s
M4	1m2.312s
Ncurses	9m0.322s
Bash	5m36.454s
Bison	2m17.604s
Bzip2	0m35.095s
Coreutils	8m24.090s
Diffutils	1m54.362s
File	0m46.442s
Findutils	1m44.260s
Gawk	3m52.328s
Gettext	34m44.299s
Grep	1m50.758s
Gzip	0m38.561s
Make	0m58.789s
Patch	1m8.531s
Perl	27m47.331s
Python	
Sed	1m21.040s
tar	3m6.694s
texinfo	3m17.785s
xz	2m40.246s

2.4 LFS 正式系统的编译

2.4.1 创建必要的文件夹，挂载必要的文件夹

注意：Debian 系统下，/dev/shm 不是/run/shm 的符号链接。

将 mount 和 chroot 命令写入一个脚本，方便出现中断时能够继续编译。

一旦 chroot 到临时系统，就无法用文本编辑器了。所以，提前写好一些脚本，方便后续使用。另外一个办法，就是另外开一个远程登录的终端，在里面写脚本。在 chroot 的系统内

执行，以减少文字的输入。

一定要注意，编译的时候要在 chroot 环境中进行，编译的时候要在 chroot 环境中进行，编译的时候要在 chroot 环境中进行！切记切记！

2.4.2 Glibc

运行 make 以后，直接 make check 的话会报错，需要

ln -sfnv $PWD/elf/ld.so.1 /lib/

Make check，会有一些失败，不过应该是正常的。

```
Summary of test results:
      132 FAIL
     5798 PASS
       16 UNSUPPORTED
       18 XFAIL
        2 XPASS
make[1]: *** [Makefile:412: tests] Error 1
make[1]: Leaving directory '/sources/glibc-2.30'
make: *** [Makefile:9: check] Error 2
```

2.4.3 Locales

调整时区，选择 China/Shanghai

2.4.4 调整工具链

2.4.5 Binutils

Make check 的时候发现了很多错误，暂时不知道这些错误有什么影响。

```
            === ld Summary ===

# of expected passes            1290
# of unexpected failures 173
# of expected failures          31
# of known failures       1
# of unresolved testcases209
# of untested testcases         2
# of unsupported tests          43
./ld-new 2.32
```

2.4.6 Gmp 6.1.2

注意在 configure 的时候增加 ABI=64

2.4.7 GCC 9.2

需要注意的有两点：

首先，修改源码配置

```
case $(uname -m) in
  x86_64)
```

```
  sed -e '/m64=/s/lib64/lib/' \
  -i.orig gcc/config/i386/t-linux64
  ;;
esac
```

上面这个脚本，尽管是针对 x86_64 处理器，但龙芯是 64 位的 MIPS 处理器，也需要进行相应的处理。如果没有处理，那么需要做一些符号链接/usr/lib64 --> /usr/lib, /lib64--> /lib。然后，修改 abi

在 configure 参数中增加

```
--with-abi=64 --with-arch=mips64r2 --with-tune=loongson3a
```

预计耗时 95 SBU，30 个小时左右。其中，编译 gcc 本身耗时 4 个小时左右，剩下的 26 个小时是在运行 make check。

实际 check 耗时 19 小时 30 分钟。

```
            === g++ Summary ===

# of expected passes            124434
# of unexpected failures 15
# of expected failures          520
# of unresolved testcases3
# of unsupported tests          6078
/sources/gcc-9.2.0/build/gcc/xg++    version 9.2.0 (GCC)
--
            === gcc Summary ===

# of expected passes            131096
# of unexpected failures 55
# of unexpected successes       8
# of expected failures          576
# of unsupported tests          3932
/sources/gcc-9.2.0/build/gcc/xgcc    version 9.2.0 (GCC)
--
            === libatomic Summary ===

# of expected passes            54
            === libgomp tests ===


Running target unix
FAIL: libgomp.c/../libgomp.c-c++-common/for-3.c execution test
--
            === libgomp Summary ===

# of expected passes            2300
# of unexpected failures 2
# of expected failures          2
```

```
# of unsupported tests          217
            === libstdc++ tests ===


--
            === libstdc++ Summary ===

# of expected passes            12642
# of unexpected failures  13
# of expected failures          78
# of unsupported tests          406

Compiler version: 9.2.0 (GCC)
```

有部分测试失败。不过好像没有什么影响，可以继续。

2.4.8 Bison

Make check 的时候报错

```
gcc: error: ./examples/c/lexcalc/scan.c: No such file or directory
gcc: fatal error: no input files
```

要避免报错，需要再装 flex

2.4.9 Libtools

5 个已知的 fail，可以无视。

```
ERROR: 140 tests were run,
65 failed (60 expected failures).
30 tests were skipped.
```

2.4.10 Libelf

Make check 有报错

不确定这个有多大的影响，先继续了。

```
Testsuite summary for elfutils 0.177
============================================================================
==========
# TOTAL: 213
# PASS:   196
# SKIP:   9
# XFAIL: 0
# FAIL:   8
# XPASS: 0
# ERROR: 0
============================================================================
==========
See tests/test-suite.log
Please report to https://sourceware.org/bugzilla
============================================================================
```

```
==========
    make[3]: *** [Makefile:2319: test-suite.log] Error 1
    make[2]: *** [Makefile:2427: check-TESTS] Error 2
    make[1]: *** [Makefile:3985: check-am] Error 2
    make: *** [Makefile:486: check-recursive] Error 1
```

2.4.11 Grub

龙芯下直接用 pmon 引导就好了，没有 grub 也没有关系。直接跳过不装了。

2.4.12 Libffi

Configure 的时候需要根据修改 gcc-arch 设置，不然会认错处理器类型。

--with-gcc-arch=native 改为--with-gcc-arch=loongson3a

测试情况如下：

```
# of expected passes        1865
# of unexpected failures    5
```

有 5 个 fail。

2.4.13 Openssl

在龙芯派下编译，需要修改一下脚本，指定编译 64 位，不然会把 mabi 指定为 n32，然后报错。

```
./Configure linux64-mips64 --prefix=/usr \
   --openssldir=/etc/ssl \
   --libdir=lib \
   shared \
   zlib-dynamic \
```

编辑器的编译可以在 ncurses 编译结束以后就进行

弄个 nano 出来，编辑脚本的时候就方便多了。

2.4.14 Gzip

Make check 的时候报错

```
estsuite summary for gzip 1.10
============================================================================
======
# TOTAL: 22
# PASS:   21
# SKIP:   0
# XFAIL: 0
# FAIL:   1
# XPASS: 0
# ERROR: 0
============================================================================
======
See tests/test-suite.log
Please report to bug-gzip@gnu.org
```

```
==================================================================
======
make[4]: *** [Makefile:1673: test-suite.log] Error 1
make[4]: Leaving directory '/sources/gzip-1.10/tests'
make[3]: *** [Makefile:1781: check-TESTS] Error 2
make[3]: Leaving directory '/sources/gzip-1.10/tests'
make[2]: *** [Makefile:2001: check-am] Error 2
make[2]: Leaving directory '/sources/gzip-1.10/tests'
make[1]: *** [Makefile:1768: check-recursive] Error 1
make[1]: Leaving directory '/sources/gzip-1.10'
make: *** [Makefile:2063: check] Error 2
```

不影响，继续

2.4.15 Patch

Make check 结果如下：

```
Testsuite summary for GNU patch 2.7.6
==================================================================
======
# TOTAL: 44
# PASS:   41
# SKIP:   1
# XFAIL: 2
# FAIL:   0
# XPASS: 0
# ERROR: 0
```

2.4.16 Tar

无法正常运行 make check。

163: storing sparse files > 8G 卡住了，系统死机。跳过 make check。

所有的软件包编译完成，耗时统计如表 3 所示，耗时超过 1 SBU 的软件包已经标红。

表 3 LFS 正式系统软件包编译耗时统计

	configure	make	make check	make install	estimated time(SBU)
man-pages				0.705s	
glibc-2.30	0m30.016s	64m1.118s	396m17.584s	7m31.277s	21
zlib	0m2.925s	0m49.244s			
file	0m50.313s	0m43.122s	0m4.382s		
readline	0m44.246s	0m39.118s		0m0.566s	
m4	2m41.530s	0m55.597s	6m57.595s	0m4.355s	0.4
bc	0m11.054s	0m33.722s	0m19.140s	0m29.928s	
binutils	0m13.748s	80m2.827s	55m41.126s	0m31.521s	7.4

gmp	1m54.931s	6m12.188s	12m56.437s	0m4.601s	1.2
mpfr	0m51.371s	4m18.668s	11m18.141s	0m3.272s	
mpc	0m28.798s	0m56.200s	3m29.325s	0m1.89s	0.9
shadow	1m15.568s	2m15.951s		0m7.841s	
gcc	0m18.659s	~230m	15h30m		95
bzip2		0m32.422s			
pkg-config	2m33.098s	3m58.009s	0m13.247s	0m3.022s	
ncurses	1m34.208s	5m42.842s			
attr	0m22.379s	0m16.364s	0m8.016s	0m2.471s	
acl	0m22.000s	0m46.798s		0m2.709s	
libcap		0m10.504s			
sed	2m26.272s	1m6.115s	3m12.908s	0m4.361s	
psmisc	0m38.149s	0m16.121s		0m1.347s	
lana-etc					
bison	5m8.140s	3m45.572s			
flex	0m54.644s	1m57.688s	6m21.637s	0m3.433s	
grep	2m32.603s	1m42.442s	3m32.100s	0m5.124s	
bash	2m27.135s	4m11.762s		0m35.122s	2.1
libtools	0m44.727s	0m24.800s	49m40.306s	0m3.342s	1.9
gdbm	0m46.257s	0m48.553s	0m30.397s	0m4.324s	
gperf	0m17.320s	0m21.224s	0m10.695s	0m0.211s	
expat	0m36.681s	0m55.862s	1m2.379s	0m2.157s	
inetutils	5m8.337s	1m50.522s	0m6.863s	0m7.372s	
Perl	7m1.604s	28m52.598s	120m46.039s	5m16.261s	9.9
XML::Parser	0m3.294s	0m36.050s	0m8.088s	0m2.374s	
intltool	0m3.057s	0m0.257s	0m20.571s		
autoconf	0m5.231s	0m8.229s	skip		
automake	0m11.512s	0m4.961s	skip		
xz	0m51.908s	1m43.972s	0m56.748s	0m6.757s	
kmod	0m44.797s	0m41.800s		0m2.316s	
gettext	8m44.025s	31m2.817s	15m2.215s		2.9
libelf	0m45.006s	10m48.053s	4m19.437s		1.1
libffi	0m36.586s	0m11.247s	8m45.260s	0m1.896s	
openssl		20m32.206s	14m53.541s		2.3
python3	3m25.361s	26m57.923s		3m5.859s	1.3
ninja		7m8.718s	4m38.574s		
meson	0m3.409s				
coreutils	6m17.050s	7m52.767s	27m47.822s	0m16.490s	2.5
check	0m54.918s	1m45.878s	3m3.321s	0m1s	
diffutils	2m27.187s	1m43.648s	2m25.192s	0m4.847s	

gawk	1m38.274s	3m49.558s	0m38.311s	0m8.039s	
findutils	5m26.908s	1m35.849s	5m18.871s	0m9.918s	0.7
groff	1m41.930s	14m4.165s		0m6.014s	0.5
grub	skip	skip	skip	skip	
less	0m31.796s	0m40.193s		0m0.118s	
gzip	1m54.839s	0m36.204s		0m14.787s	
iproute		3m58.757s		0m0.861s	
kdb	0m45.642s	1m59.897s	0m5.913s	0m7.376s	
libpipeline	1m22.228s	0m32.005s	0m15.963s	0m2.356s	
make	0m50.358s	0m54.207s	1m10.733s	0m1.690s	
patch	2m56.943s	1m3.018s	0m11.971s	0m1.934s	
man-DB	3m30.067s	2m52.653s	0m51.207s	0m9.619s	
tar	3m34.007s	3m1.282s			2.2
texinfo	2m34.673s	3m56.241s	7m19.788s	0m7.184s	0.8
vim	1m24.436s	12m33.625s	skip	0m6.271s	2.2
procps-ng	0m57.856s	1m44.800s	0m18.436s	0m5.378s	
utils-linux	1m56.788s	12m36.862s	skip	0m28.059s	1.2
e2fsprogs	1m43.943s	7m4.802s	5m24.590s	0m9.720s	3.1
sysklog		0m11.004s			
sysvinit		0m16.890s		0m0.519s	
eudev	1m2.548s	2m39.424s	0m12.926s	0m6.798s	
nano	2m26.059s	2m44.547s			

2.4.17 后续配置

网络部分

在/etc/sysconfig 文件夹中，增加一个文件 ifconfig.enp0s31f1：

```
ONBOOT=yes
IFACE=enp0s31f1
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
```

这个文件定义了第二个网口的设置。如果需要使用第一个网口，可以自行修改。

/etc/sysconfig/console

文件不要乱写。我犯了一个错误，把 KEYMAP 设置成了 de-latin1，结果键盘被当成了德语键盘，键盘的 Z 和 Y 被交换了位置。

建议直接留个空文件，或者 KEYMAP 设置为 US。

为了支持串口登录，在/etc/inittab 文件中增加一行：

```
S1:12345:respawn:/sbin/agetty -L 115200 ttyS0 vt100
```

这样开机以后串口也有一个登录的界面。

内核编译部分直接跳过，继续用 4.19 内核。有必要的话，可以自行编译一个更合适的内核。

配置好了，重启，在 pmon 界面选择 LFS 系统，设置无误的话就可以正常进入系统了！

刚编译完成的 LFS 只包含最最基本的包，距离生产力还差得很远。比如，连 ssh 远程登录都没有。

2.4.18 BLFS 包安装

参考 BLFS 手册，进行设置。

http://linuxfromscratch.org/blfs/view/stable/

安装 Nano

文档：http://linuxfromscratch.org/blfs/view/stable/postlfs/nano.html

Nano 依赖 ncurses。编译完 ncurses，就可以安装 nano。

安装 openssh

文档：http://linuxfromscratch.org/blfs/view/stable/postlfs/openssh.html

Openssh 依赖 openssl。

注意需要下载 BLFS 脚本

http://anduin.linuxfromscratch.org/BLFS/blfs-bootscripts/blfs-bootscripts-20190609.tar.xz

安装 wget

文档：http://linuxfromscratch.org/blfs/view/stable/basicnet/wget.html

Dhcp（可选，我还没有安装）

用于自动获取 IP。这样就不需要自己手工设置 IP 了。

http://linuxfromscratch.org/blfs/view/stable/basicnet/dhcpcd.html

3. LFS 编译过程的个人经验和建议

3.1 经验

LFS 编译很繁琐，建议做好笔记，做个表格，记录一下自己做到哪一步了。龙芯派的性能比较有限，双核 1GHz，整体的运行速度可能只有主流 X86 多核 CPU 的 1/10,因此构建 LFS 耗时较多，不能在一天之内完成。

3.2 有可能犯的错误

(1) 少敲命令，多敲命令，拼写错误。比如，make check 以后忘记 make install。没有什么好的解决办法，多注意。

(2) 忘记在 chroot 环境中编译。在 chroot 环境中，连 vim 都没有，编辑文件都不方便，所以，我会同时开两个终端，第一个没有 chroot 的终端下编辑 LFS 用的脚本，第二个终端在 chroot 下编译 LFS。经常会在第一个终端下编辑完脚本以后顺手就跑了一遍，发现错误以后再重新在 chroot 终端下跑一遍。为了防止出现这个错误，个人建议是在 chroot 环境中编译完成 ncurses 以后就立即编译一个 nano 或者 vim 编辑器，然后直接在 chroot 环境中编辑必要的文件，避免犯错。