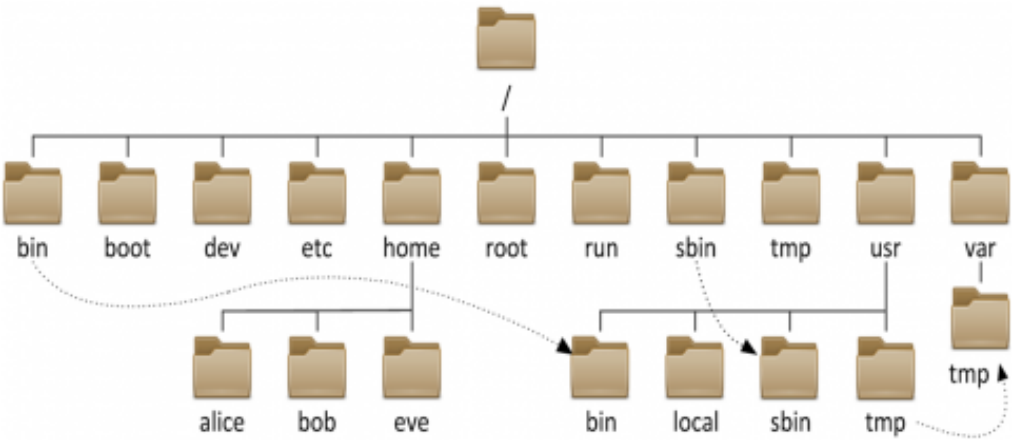
Linux目录结构



目录结构特点

- 采用级层式的树状目录结构，以 `/` 为根目录
- 文件大小写敏感
- 以 `.` 开头的是隐藏文件
- 文件名最长255字节
- 包括路径在内的文件名最长4095个字节
- 蓝色：目录，绿色：二进制文件，红色：压缩文件，浅蓝色：链接，灰色：其他文件
- 有些字符需要使用引号来使用
- 每个文件都有两类相关数据：元数据：metadata，也就是属性，和数据：data，也就是文件内容

常见目录及其功能

- `/bin` ：Binary的缩写，存放二进制文件，类似的还有 `/usr/bin` 和 `/usr/local/bin`
- `/sbin` ：Super User Binary的意思，存放的是系统管理员使用的系统管理程序
- `/home` ：普通用户主目录，以 `/home/ozliinex` 为例，每个用户都有自己的目录
- `/root` ：root用户主目录，单独出来的
- `/lib` ：系统开机所需的最基本动态链接共享库，几乎所有应用程序都需要这些共享库
- `/lost+found` ：一般是空的，非法关机后这里就存放一些文件
- `/etc` ：所有的系统管理所需要的配置文件和子目录，以Nginx为例，这里会有一个 `/etc/nginx` 目录，里面可以配置Nginx的配置
- `/usr` ：用户的应用程序与文件均存放于这个目录下
- `/boot` ：启动Linux的一些核心文件，包括一些链接文件和镜像
- `/proc` ：虚拟目录，这是系统内存的映射，访问这个目录来获取系统信息
- `/srv` ：service的缩写，存放一些服务启动后需要提取的数据
- `/sys` ：用于将系统中的设备组织成层次结构，并向用户模式程序提供详细的内核数据结构信息
  - `/sys/devices` ：全局设备结构体系，包含所有被发现的注册在各种总线上的各种物理设备
  - `/sys/dev` ：存放主次设备号文件

    - `/sys/class`：包含所有注册在Kernel里面的设备类型，按照设备功能分类的设备模型，每个设备类型表达具有一种功能的设备
    - `/sys/bus`：总线嘛，这个目录下每个子目录都是Kernel支持并且已经注册的总线类型
    - `/sys/power`：系统中的电源选项，这个目录下有几个文件可以用于控制整个机器的电源状态，比如可以向里面写入控制命令让机器关机/重启
- `/tmp`：存放临时文件
- `/mnt`：让用户临时挂载别的文件系统，如可以把外部存储挂载在 `/mnt` 上
- `/media`：Linux系统会自动识别一些设备，例如U盘、光驱等，识别后，Linux会把识别的设备挂载到这个目录下
- `/opt`：给主机额外安装软件所存放的目录，例如安装 `ORACLE`
- `/usr/local`：另一个给主机额外安装软件所安装的目录，一般通过编译源码方式安装
- `/var`：存放在不断扩充的东西，习惯将经常被修改的目录放在这个目录下。包括各种日志文件
- `/selinux`：SELiunx是一种安全子系统（沙箱？），它能控制程序只能访问特定文件，有三种工作模式。这是selinux相关的安全策略等信息的存储位置

HPL.dat

完整代码

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
4            # of problems sizes (N)
29 30 34 35  Ns
4            # of NBs
1 2 3 4      NBs
0            PMAP process mapping (0=Row-,1=Column-major)
3            # of process grids (P x Q)
2 1 4        Ps
2 4 1        Qs
16.0         threshold
3            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
2            # of recursive stopping criterium
2 4          NBMINs (≥ 1)
1            # of panels in recursion
2            NDIVs
3            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (≥0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
```

```
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

参数分析

第5-9行

`4 # of problems sizes (N)`：矩阵数量

`29 30 34 35 Ns`：问题规模大小N值，要考虑内存容量的制约关系，有一个达到最佳性能的上限制。可以利用物理内存的容量(Byte)的80-85%来进行HPL运算，剩余内存保证系统正常运行。由于一个双精度数占8个字节，因此再除以8，将结果开平方，得到比较接近的最佳N值。经验上似乎为384的倍数更佳。

`4 # of NBs`：测试块的个数

`1 2 3 4 NBs`：系数矩阵被分成NB×NB的循环块，分配到各个进程当中去处理，NB大小作为计算粒度，在很大程度上影响了计算性能的优劣。NB不可能太大或太小，一般在256以下，NB×8一定是Cache Line的倍数等。一般通过单节点或单CPU测试可以得到几个较好的NB值，但当系统规模增加、问题规模变大，有些NB取值所得性能会下降。所以最好再小规模测试时选3个左右性能不错的NB值，再通过大规模测试检验这些选择。此处一般选择128。

`0 PMAP process mapping (0=Row-,1=Column-major)`：选择处理器阵列是按列还是按行排列。按列的适用于节点数较多、每个节点内CPU数较少的系统；而按行的排列方式适用于节点数较少、每个节点内CPU数较多的大规模系统。在集群系统上，按照列的排列方式性能远好于按行的排列方式。此处一般选择1

第10-12行

`3 # of process grids (P x Q)`：

这一行与下面的 `2 1 4 Ps` 和 `2 4 1 Qs` 相关，用来说明二维处理器网格（P×Q），二维处理器网格有这些要求：

- P×Q = 系统CPU数 = 进程数
- 一般来说一个进程对于一个CPU可以得到最佳性能
- 对于Intel Xeon来说，关闭超线程可以提高HPL性能
- P≤Q，一般来说P尽量取小一点，因为列项通信（通信次数和通信数据量）要远大于横向通信。
- P=2^n，P最好选择2的幂。
- HPL中，L分解的列向通信采用二元交换法（Binary Exchange），当列向处理器个数P为2个幂时，性能最优
- 例如，当系统进程数为4，且问题规模较大时，P×Q选择为1×4的效果要比选择2×2好一些，但当问题规模较小时，二者差距并不大，因为此时节点内的计算开销比通信开销大很多，所以网格的分布方式对整个性能的影响就比较小了。在集群测试中，P×Q=系统CPU总核数

第13行

`16.0 threshold`：测试精度，一般来说没必要修改这个值

第14-21行

这部分指明了L分解的方式，在消元中，zHPL采用每次完成NB列的消元，然后更新后面的矩阵。这NB的消元就是L的分解。每次L的分解只在一列处理器中完成。对每一个小矩阵作消元时，都有3种算法：L(Left)、R(Right)、C(Crout)。在LU分解中，具体的算法很多。

测试经验，NDIVs选择2比较理想，NBMINs 4或8都不错。而对于RFACTs和PFACTs，对性能的影响不大。

在HPL官方文档中，推荐的设置为：

```
1       # of panel fact
1       PFACTs (0=left, 1=Crout, 2=Right)
2       # of recursive stopping criterium
4 8     NBMINs (>= 1)
1       # of panels in recursion
2       NDIVs
1       # of recursive panel fact.
2       RFACTs (0=left, 1=Crout, 2=Right)
```

第22-23行

这部分说明了L的横向广播方式，HPL中提供了6种广播方式。其中，前4种适合于快速网络；后两种采用将数据切割后传送的方式，主要适合于速度较慢的网络。目前，机群系统一般采用千兆以太网甚至光纤等高速网络，所以一般不采用后两种方式。

一般来说，在小规模系统中，选择0或1；对于大规模系统，选择3。

推荐的配置为

```
2            # of broadcast
3            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
```

第24-25行

这部分说明了横向通信的通信深度。这依赖于机器的配置和问题规模的大小

推荐配置为

```
2            # of lookahead depth
0 1          DEPTHs (>=0)
```

26-27行

这部分说明U的广播算法。

U的广播为列向广播，HPL提供了3种U的广播算法：二元交换（Binary Exchange）法、Long法和二者混合法。

SWAP0、1、2分别对应二元交换法、Long法、混合法

推荐配置为

```
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
```

第28-29行

这部分说明L和U的数据存放格式。

若选择"transposed"，则采用按列存放，否则按行存放。

推荐配置为

```
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
```

30-31行

- `1            Equilibration (0=no,1=yes)`：一般在回代中使用，一般使用其默认值
- `8            memory alignment in double (> 0)`：主要为内存地址对齐而设置，用于在内存分配中对齐地址。出于安全考虑，可以选择8