1 开发环境配置

1.1 前提条件

在进行C程序开发前需安装好操作系统，具体安装步骤请参考操作系统安装文档。

1.2 网络设置

1.2.1 配置网络

在统信服务器操作系统中，一般使用配置网络接口的命令行工具：ifconfig、ifup、ifdown。

1.2.2 ifconfig命令

管理员用户可以使用ifconfig命令查看网络设备状态信息以及对网络接口进行配置。

■ 命令格式：ifconfig [网络设备] [参数]

■ 常用参数

表1.1 ifconfig参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| add <地址> | 设置网络设备的IP地址。 |
| del <地址> | 删除网络设备的IP地址。 |
| down | 关闭指定的网络设备。 |
| netmask <子网掩码> | 设置网络设备的子网掩码。 |

说明：在终端界面，执行man ifconfig，查看其他更多参数及其含义

■ 执行示例

◆ 执行 ifconfig ，显示网络设备信息。

◆ 执行 ifconfig eth1 down，关闭网络接口eth1。

◆ 执行 ifconfig -a ，显示全部当前有效的接口信息。

◆ 执行 ifconfig eth1 192.168.1.88 netmask 255.255.255.0，设置网络接口eth1的ip地址，子网掩码。

1.2.3 ifup命令

管理员用户可以使用ifup命令激活一个指定的网络接口。

执行 ifup eth1，激活网络接口eth1。

1.2.4 ifdown命令

管理员用户可以使用ifdown命令关闭一个指定网络接口。

执行 ifdown eth1，关闭网络接口。

1.2.5 管理IP

统信服务器操作系统中，一般使用管理IP的命令行工具：ip、dhclient、route。

ip命令

ip命令用来显示或操纵Linux主机的路由、网络设备、策略路由和隧道，是Linux下较新的功能强大的网络配置工具。

使用ip命令，只需一个命令，你就能很轻松地执行一些网络管理任务。相对于ifconfig，ip命令功能更强大。

统信服务器操作系统已经预装了iproute，可以轻松的设置IP。

■ 命令格式：ip [参数] OBJECT { COMMAND | help }

■ 常用参数

表1.2 ip参数表

|  |  |
| --- | --- |
| 常用参数 | 对象说明 |
| link | 网络设备。 |
| address | 设备上的协议（IP或IPv6）地址。 |
| addrlabel | 协议地址选择的标签配置。 |
| route | 路由表条目。 |
| rule | 路由策略数据库中的规则。 |
| -V | -Version：显示指令版本信息。 |
| -s | -stats，statistics：输出详细信息。 |
| -h | -human，-human-readable：输出人类可读的统计信息和后缀。 |
| -o | -oneline：将每条记录输出到一行，用‘\’字符替换换行符。 |

■ 执行示例

◆ 执行 ip addr show，或 ip address show，显示网卡及配置的地址信息。

◆ 执行 ip addr add 192.168.0.123/24 dev eth1，设置IP地址。24表示子网掩码是24表示子网掩码的长度是24位，换算成十进制是255.255.255.0。

◆ 执行 ip addr del 192.168.0.123/24 dev eth1，删除ip地址。

◆ 执行 ip link set eth1 up ，启用被禁用的网卡eth1。

◆ 执行 ip link set eth1 down，禁用网卡。

◆ 执行 ip -s link list，显示详细的设备信息。

◆ 执行 ip route list，显示核心路由表。

dhclient命令

dhclient命令使用动态主机配置协议动态的配置网络接口的网络参数。

■ 命令格式：dhclient [参数]

■ 常用参数

表1.3 dhclient参数表

|  |  |
| --- | --- |
| **参数** | **说明** |
| -p | 指定dhcp客户端监听的端口号。 |
| -d | 总是以前台方式运行程序。 |
| -r | 释放ip地址。 |

■ 执行示例

◆ 执行 dhclient -4 eth0指定dhclient支持ipv4协议。

◆ 执行 dhclient -r，释放当前IP。

route命令

Linux系统的route命令用于显示和操作IP路由表，要实现两个不同的子网之间的通信，需要一台连接两个网络的路由器，或者同时位于两个网络的网关来实现。

备注：直接在命令行下执行route命令来添加路由，不会永久保存，当网卡重启或者机器重启之后，该路由就失效了；可以在/etc/rc.local中添加route命令来设置路由永久有效。

■ 命令格式：route [参数]

■ 常用参数

表1.4 route参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -n | 不解析名字 |
| -v | 显示详细的处理信息 |
| -F | 显示发送信息 |
| -C | 显示路由缓存 |
| add | 添加一条新路由。 |
| del | 删除一条路由。 |
| -net | 目标地址是一个网络。 |
| -host | 目标地址是一个主机。 |
| netmask | 当添加一个网络路由时，需要使用网络掩码。 |
| metric | 设置路由跳数。 |
| gw | 路由数据包通过网关。注意，你指定的网关必须能够达到。 |

■ 执行示例

◆ 执行 route add -net 192.168.2.0 netmask 255.255.255.0 dev eth0， 添加一条到达192.168.2.0网络的路由，指定网络掩码为255.255.255.0，数据包通过网络接口eth0。

◆ 执行 route add -net 192.168.66.0 netmask 255.255.255.0 gw 192.168.2.1， 添加一条到达192.168.66.0网络的路由，指定网络掩码为255.255.255.0，数据包通过网关地址192.168.2.1。

◆ 执行 route add -host 192.168.66.200 gw 192.168.2.1，所有去往192.168.66.200主机的数据包发往网关地址192.168.2.1。

◆ 执行 route add default gw 192.168.1.1 ，添加一条默认网关，所有的数据包将被转发到192.168.1.1。

◆ 添加或设置网关，执行route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0命令。

◆ 删除路由记录，执行route del -net 224.0.0.0 netmask 240.0.0.0命令。

1.2.6 网络诊断

ping命令

ping命令是常用的网络命令，也属于一个通信协议，是TCP/IP协议的一部分。使用ping命令可以检查网络是否连通，能够帮助管理员分析和判定[网络故障](http://baike.baidu.com/view/76955.htm)。

执行ping指令会使用ICMP传输协议，发出要求回应的信息，若远端主机的网络功能没有问题，就会回应该信息，因而得知该主机运作正常。ICMP协议是Internet控制报文协议。是TCP/IP协议簇的一个子协议，用于在IP主机、路由器之间传递控制消息。

■ 命令格式：ping [参数] 地址

■ 常用参数

表1.5 ping参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -d | 使用Socket的SO\_DEBUG功能。 |
| -c | <完成次数> 设置完成要求回应的次数。 |
| -f | 极限检测。 |
| -i | <间隔秒数> 指定收发信息的间隔时间。 |
| -l | <前置载入> 设置在送出要求信息之前，先行发出的数据包。 |
| -n | 只输出数值。 |
| -p | <范本样式> 设置填满数据包的范本样式。 |
| -q | 不显示指令执行过程，开头和结尾的相关信息除外。 |
| -r | 忽略普通的Routing Table，直接将数据包送到远端主机上。 |
| -R | 记录路由过程。 |
| -s | <数据包大小> 设置数据包的大小。 |
| -t | <存活数值> 设置存活数值TTL的大小。 |

■ 执行示例

◆ 执行 ping 192.168.0.12，查看本机的网络状态。

◆ 执行 ping -i 3 -s 1024 -t 255 www.baidu.com，-i 3 发送周期为 3秒， -s 设置发送包的大小 -t 设置TTL值为 255。

◆ 执行 ping -c 2 www.baidu.com，收到两次包后，自动退出。

◆ 执行 ping -i 0.5 -c 10 www.baidu.com，每0.5秒发送一次网络封装包并限制发送10次。

traceroute命令

通过traceroute我们可以知道信息从你的计算机到互联网另一端的主机是走的什么路径。当然每次数据包由某一同样的出发点（source）到达某一同样的目的地(destination)走的路径可能会不一样，但基本上来说大部分时候所走的路由是相同的。

traceroute主要用于追踪网络数据包的路由途径，默认值数据包大小是40Bytes，用户可自定义大小。

注意：在统信服务器操作系统中，默认未集成traceroute，手动执行yum install traceroute，安装即可使用。

■ 命令格式：traceroute [参数] [地址]

■ 常用参数

表1.6 traceroute参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -d | 使用Socket层级的排错功能。 |
| -f | 设置第一个检测数据包的存活数值TTL的大小。 |
| -F | 设置勿离断位。 |
| -g | 设置来源路由网关，最多可设置8个。 |
| -i | 使用指定的网络界面送出数据包。 |
| -I | 使用ICMP回应取代UDP资料信息。 |
| -m | 设置检测数据包的最大存活数值TTL的大小。 |
| -n | 直接使用IP地址而非主机名称。 |
| -p | 设置UDP传输协议的通信端口。 |
| -r | 忽略普通的Routing Table，直接将数据包送到远端主机上。 |
| -s | 设置本地主机送出数据包的IP地址。 |
| -t | 设置检测数据包的TOS数值。 |
| -w | 设置等待远端主机回报的时间。 |

■ 执行示例

◆ 执行 traceroute www.baidu.com，显示到达目的地的数据包路由。

◆ 执行traceroute -m 4 www.baidu.com，显示到达目的地的数据包路由，设置数据存活数值为4。

1.2.7 域名管理

在统信服务器操作系统中，一般使用管理域名的命令行工具：host、nslookup。

host命令

host 是一个常用的DNS查询工具，经常用来查询域名、检查域名解析是否正确。

■ 命令格式：host [参数] name [server]

■ 常用参数

表1.7 host参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -a | 查询所有的信息。 |
| -c | 设置查询类型。 |
| -C | 查询完整的SOA记录。 |
| -d | -v | 显示详细过程。 |
| -l | 列表模式。 |
| -t | 选择查询类型：CNAME NS SOA SIG KEY AXFR。 |
| -w | 永久等待。 |
| -W | 设置等待超时。 |

■ 执行示例：执行 host -a www.baidu.com，查看域名地址的详细信息。

nslookup命令

nslookup是一个用于查询 [Internet](http://baike.baidu.com/view/11165.htm)[域名](http://baike.baidu.com/view/43.htm)信息或诊断DNS服务器问题的工具。主要用来诊断[域名系统](http://baike.baidu.com/view/124777.htm)的基础信息。

■ 命令格式：nslookup [参数] [dns-server]

■ 执行示例

◆ 执行 nslookup www.baidu.com ，查询网站域名信息。

◆ 执行 nslookup -q=mx mail.qq.com，查看邮件服务器的信息。

1.2.8 网络状态

netstat命令

在统信服务器操作系统中，一般通过netstat命令来查看网络连接状态，当设置不同命令参数时，可以显示网络连接状态、路由表、接口状态、无效连接和多播成员。也可以用于查看某个服务是否运行。

■ 命令格式：netstat [参数]

■ 常用参数

表1.8 netstat参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -a | (all)显示所有选项，netstat默认不显示LISTEN相关。 |
| -t | (tcp)仅显示tcp相关选项。 |
| -u | (udp)仅显示udp相关选项。 |
| -n | 拒绝显示别名，能显示数字的全部转化成数字。(重要) |
| -l | 仅列出有在 Listen (监听) 的服务状态。 |
| -p | 显示建立相关链接的程序名。 |
| -r | 显示路由信息，路由表。 |
| -e | 显示扩展信息，例如uid等。 |
| -s | 按各个协议进行统计。(重要) |
| -c | 连续打印所选信息。 |

说明：在终端界面，执行man netstat ，查看其他更多参数及其含义

■ 执行示例

◆ 执行 netstat -a，列出所有网路端口。

◆ 执行 netstat -i，显示网卡列表。

◆ 执行 netstat -apu ，查看UDP端口的使用情况。

◆ 执行 netstat -s，显示网络统计信息。

1.3 用户管理

用户管理常用的功能：添加用户、修改用户、删除用户、口令管理、添加用户组、修改用户组、删除用户组。

1.3.1 添加用户

useradd命令用来创建或更新用户信息。

■ 命令格式：useradd [参数] 用户名

■ 常用参数

表1.9 useradd参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -c | comment给新用户添加备注。 |
| -d | home\_dir为主目录指定一个名字（如果不想用登陆名作为主目录名的话）。 |
| -m | 创建用户的HOME目录。 |
| -M | 不创建用户的HOME目录（当默认设置里指定创建时，才用到）。 |
| -N | 不要创建与用户同名的组。 |
| -r | 创建系统用户。 |
| -p | passwd为用户指定默认密码。 |
| -s | shell指定默认登陆shell。 |
| -u | uid为帐户指定一个唯一的UID。 |
| -U | --user-group创建与用户同名的组。 |

说明：执行man useradd命令，查看其他更多参数及其含义。

■ 执行示例：以root用户身份登陆系统，打开终端界面。

◆ 执行useradd testuser1添加用户testuser1。

◆ 执行useradd -m testuser2命令，创建用户主目录。

◆ 执行useradd -m -s /bin/bash    testuser1命令，指定用户登陆后使用的shell。

◆ 执行useradd -U testuser2命令，创建一个和用户同名的组，并将用户添加到组中。

说明：

■ 执行less /etc/passwd命令，可以查看新增加用户testuser1的信息。

■ 执行cat /etc/shells可以查看系统安装了哪些shell。

1.3.2 修改用户

usermod命令可用来修改用户帐号的各项设定。

■ 命令格式：usermod [参数] 用户名

■ 常用参数

表1.10 usermod参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -a | --append 把用户追加到某些组中，仅与-G一起使用。 |
| -d | --home用户的新登陆目录。如果给了-m选项，当前主目录的内容会移动到新主目录中，如果不存在，则创建。 |
| -l | --login用户的名称将会从原用户修改为新用户。 |
| -G | --groups GROUP1[,GROUP2,...[,GROUPN]]]用户属于的附加组列表。组之间使用逗号分隔，没有空格。 |
| -u | --uid UID设置用户ID的新数值。 |

说明：执行man usermod命令，查看其他更多参数及其含义。

■ 执行示例

◆ 执行usermod -aG testgrp testuser多个组之间用空格隔开，把testuser用户加入testgrp组。可以使用id testuser查看是否追加到组成功。

说明：执行id testuser命令，查看用户加入的组。

◆ 执行usermod -l testuser testuser1命令，修改testuser1的用户名为testuser。

◆ 执行usermod -d /home/testuser4 -m testuser命令，修改用户名为testuser的home名。

◆ 执行usermod -u 10002 testuser命令，修改用户名为testuser的UID。

1.3.3 删除用户

■ 命令格式：userdel [参数] 用户名

■ 常用参数

表1.11 userdel参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -h | --help显示帮助信息。 |
| -r | --remove用户主目录中的文件将随用户主目录和用户邮箱一起删除。在其它文件系统中的文件需要手动搜索并删除。 |
| -Z | --selinux-user为用户删除所有的SELinux用户映射。 |

说明：执行man userdel命令，查看其他更多参数及其含义。

■ 执行示例：以root用户身份登陆系统，打开终端界面。

◆ 执行userdel testuser1命令，将删除testuser1相关的用户数据文件，如 /etc/passwd、/etc/group、/etc/shadow等都将被删除。但不删除home目录及文件。

◆ 执行userdel -r testuser1命令，删除用户目录及其它相关文件，home目录及文件也将删除。

注意：请不要轻易用-r参数；他会删除用户的同时删除用户所有的文件和目录，切记如果用户目录下有重要的文件，在删除前请备份。

1.3.4 口令管理

■ 命令格式：passwd [参数] 用户名

■ 常用参数

表1.12 passwd参数表

|  |  |
| --- | --- |
| **参数** | **描述** |
| -l | 锁定口令，即禁用帐号。 |
| -u | 解锁指定用户的密码。 |
| -d | 删除用户密码，这是禁用一个用户密码的快速方法。 |

说明：执行man passwd命令，查看其他更多参数及其含义

■ 执行示例

◆ 执行passwd testuser3，修改testuser3这个用户的密码。输入新密码，确认新密码。

◆ 执行passwd -l testuser3，锁定testuser3这个用户。

◆ 执行passwd -u testuser3，解锁testuser3这个用户。

说明：远程终端，执行ssh testuser3@192.168.0.20，输入密码，验证操作是否生效。192.168.0.20是服务器的IP地址。

■ 认证模块：passwd具体调用的认证模块与/etc/pam.d/system-auth 和/etc/pam.d/password-auth 中的相关配置有关，配置细节请查看PAM章节常用模块介绍pam\_pwquality.so和pam\_deepin\_pw\_check.so。

1.4 仓库设置

讲述如何配置操作系统软件仓库，使操作系统可以通过yum/dnf等命令进行在线安装、卸载或升级应用软件开发所需要的软件包。

在统信服务器操作系统安装完成后，首先对软件源配置文件进行配置并升级。简单来说就是你安装软件时，程序从哪个仓库地址获取软件包。

1.4.1 配置软件源

软件源的配置一般有两种方式，一种是直接配置 /etc/dnf/dnf.conf 文件，另外一种是在 /etc/yum.repos.d 目录下增加 .repo 文件。

/etc/dnf/dnf.conf 文件包含的“main”部分，配置文件示例如下：

**[main]**

gpgcheck=0

installonly\_limit=3

clean\_requirements\_on\_remove=True

best=True

Skip\_if\_unavailable=False

常用选项说明：

表1.13 repo配置参数表

|  |  |
| --- | --- |
| **参数** | **说明** |
| cachedir | 缓存目录，该目录用于存储RPM包和数据库文件。 |
| keepcache | 可选值是1和0，表示是否要缓存已安装成功的那些RPM包及头文件，默认值为0，即不缓存。 |
| debuglevel | 设置dnf生成的debug信息。取值范围：[0-10]，数值越大会输出越详细的debug信息。默认值为2，设置为0表示不输出debug信息。 |
| clean\_requirements\_on\_remove | 删除在dnf remove期间不再使用的依赖项，如果软件包是通过DNF安装的，而不是通过显式用户请求安装的，则只能通过clean\_requirements\_on\_remove删除软件包，即它是作为依赖项引入的。 默认值为True。 |
| best | 升级包时，总是尝试安装其最高版本，如果最高版本无法安装，则提示无法安装的原因并停止安装。 默认值为True。 |
| obsoletes | 可选值1和0，设置是否允许更新陈旧的RPM包。默认值为1，表示允许更新。 |
| gpgcheck | 可选值1和0，设置是否进行gpg校验。默认值为1，表示需要进行校验。 |
| plugins | 可选值1和0，表示启用或禁用dnf插件。默认值为1，表示启用dnf插件。 |
| installonly\_limit | 设置可以同时安装“installonlypkgs”指令列出包的数量。默认值为3，不建议降低此值。 |

repository部分允许您定义定制化的软件源仓库，各个仓库的名称不能相同，否则会引起冲突。下面是[repository]部分的一个最小配置示例：

**[repository]**

name=repository\_name

baseurl=repository\_url

repository参数说明：

表1.14 repo参数表

|  |  |
| --- | --- |
| **参数** | **说明** |
| name=repository\_name | 软件仓库（repository ）描述的字符串 |
| baseurl=repository\_url | 软件仓库（repository ）的地址。■ 使用http协议的网络位置：例如<http://path/to/repo>■ 使用ftp协议的网络位置：例如<ftp://path/to/repo>■ 本地位置：例如file:///path/to/local/repo |

统信服务器操作系统中有默认的repo文件UniontechOS.repo，其中包含AppStream、BaseOS、PowerTools、Plus、Update等源。

UnionTechOS.repo文件的AppStream配置如下所示：

**[UniontechOS-$releasever-AppStream]**

name = UniontechOS $releasever AppStream

baseurl= https://enterprise-c-packages.chinauos.com/server-enterprise-c/kongzi/1050/AppStream/$basearch

enabled = 1

username=$auth\_u

password=$auth\_p

gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-uos-release

gpgcheck = 0

skip\_if\_unavailable = 1

说明：

■ UnionTechOS默认已经配置好了镜像源仓库，需要激活授权后才能使用。

■ UnionTechOS默认配置中有认证相关的配置，请勿修改repo默认配置文件，否则会导致软件包更新失败,username和password为认证所需要的用户名和密码。

1.4.2 安装软件包

■ 执行以下命令安装特定软件包。

dnf install PKGNAME

■ 执行以下命令更新所有已安装的软件包。

dnf upgrade

注意：更新软件包将被下载到/var/cache/dnf/目录下缓存，然后再进行安装。升级的记录会写在日志文件/var/log/dnf.log、/var/log/dnf.librepo.log、/var/log/dnf.rpm.log中。

1.5 开发工具安装

1.5.1 代码编辑工具安装

讲述vim等软件工具的安装

在安装软件前需要完成系统软件仓库的配置，软件仓库具体配置方法参考“仓库设置”章节的描述。

Linux系统图像化模式下，系统默认会安装vi、vim等编译关键，在Linux系统最小化安装或docker环境中，可以使用yum install vim命令来安装vim。

1.5.2 代码调试工具安装

在安装软件前需要完成系统软件仓库的配置，软件仓库具体配置方法参考“仓库设置”章节的描述。

gdb：调试器

Valgrind：是一款用于内存调试、内存泄漏检测以及性能分析的软件开发工具。

ltrace：用来跟踪进程调用库函数的情况；

strace：监控用户空间进程和内核的交互,比如系统调用、信号传递、进程状态。

通过yum install命令来安装gdb valgrind ltrace strace。

yum install gdb valgrind ltrace strace

1.5.3 版本管理工具安装

在安装软件前需要完成系统软件仓库的配置，软件仓库具体配置方法参考“仓库设置”章节的描述。

git和svn都是版本控制管理工具，通过采用分支管理系统的高效管理，实现共享资源，实现最终集中式的管理。

通过yum install命令来安装git和svn。

1.6 虚拟化环境配置

配置虚拟化需要硬件的支持，需要CPU支持虚拟化，Linux系统下可以通过cat /proc/cpuinfo | grep vmx或cat /proc/cpuinfo | grep smx命令来查看。

在安装软件前需要完成系统软件仓库的配置，软件仓库具体配置方法参考“仓库设置”章节的描述。

虚拟化相关软件属于“virt”module，可以通过module形式安装。执行dnf module install virt即可完成安装。

1.7 容器环境配置

     在安装软件前需要完成系统软件仓库的配置，软件仓库具体配置方法参考“仓库设置”章节的描述。

虚拟化相关软件属于“container-tools”module，可以通过module形式安装。执行dnf module install container-tools即可完成安装。

1.8 编译环境配置

在开始开发C程序前，还需要完成基本开发库和相关编译器的安装部署。例如：

通过dnf module install llvm-toolset命令安装clang开发工具集；

通过dnf install glibc glibc-devel glibc-headers命令安装glib相关库和头文件；

通过dnf install kernel-headers命令安装kernel头文件；

通过dnf module install mariadb mariadb-devel 命令安装mariadb数据库和数据库相关库文件；

通过dnf module install virt-devel命令安装虚拟化相关开发库和头文件。

2 程序开发

2.1 编译C程序

预编译：使用-E参数可以让GCC在预处理结束后停止编译过程

gcc -E hello.c -o hello.i

预处理的宏定义插入到hello.i中

生成目标代码：将hello.i编译为目标代码，通过使用-c参数来完成

gcc -c hello.i -o hello.o

也可以通过源文件直接生成

gcc -c hello.c

生成可执行文件：

gcc hello.o -o hello

也可以通过源文件直接生成

gcc -o hello hello.c

2.2 使用C库

C库文件使用so或a为后缀，一个名为foo库的通常动态库为 libfoo.so，静态库为libfoo.a。

当使用gcc在编译一个C程序是，可以使用“-l”选项指定要依赖的库，在使用库是可用通过库名称来指定，不需要加后缀。如：gcc ...... -lfoo

当在生成一个库时，需要指定全名，如：gcc ...... -o libfoo.so.x.x

如果在某个目录下有编译时所需要的头文件，为了让GCC能够顺利地找到它们，就可以使用-L选项，如：gcc ...... -L /usr/local/lib/

2.3 创建C库

2.3.1 生成静态库

在完成了C程序编译后，执行下面命令将所有生成的.o文件打包生成静态库。

ar rcs libfoo.a \*.o

2.3.2 生成动态库

生成与位置无关的.o文件

gcc -fPIC \*.c -c

将生成的所有.o文件生成动态库

gcc -shared -o libfoo.so \*.o

2.4 使用make管理多个程序

   当一个软件是由多个C程序文件组成，可以通过Makefile来管理编译，例如一个软件有main.c、sub.c、add.c、add\_sub.h 4个文件，避免每次重复输入繁琐的编译命令，可以通过编写makefile文件后执行make命令来解决。

   创建makefile文件，内容如下：

main: add.o sub.o main.o

   gcc add.o sub.o main.o -o main

add.o: add.c add\_sub.h

   gcc -c add.c -o add.o

sub.o: sub.c add\_sub.h

   gcc -c sub.c -o sub.o

main.o:main.c add\_sub.h

   gcc -c main.c -o main.o clean:

rm -rf \*.o main

3 调试程序

3.1 启用程序debug

如果要使用gdb调试程序，需要在使用gcc编译c程序时使用-g选项开启程序debug模式。

3.2 使用GDB调试程序

在使用gdb调试程序前，需要使用yum install gdb命令安装gdb，gdb安装后，使用gdb appname进行调试程序，appname为实际要调试的程序。

gdb常用的子命令有设置断点、查看变量、重启/退出调试等等。

run(r) ： 重新开始运行文件（run-text：加载文本文件，run-bin：加载二进制文件）

run argv[1] argv[2]：调试时命令行传参

start ：单步执行，运行程序，停在第一行执行语句

next(n) ：单步调试（逐过程，函数直接执行）

step(s)：单步调试（逐语句：跳入自定义函数内部执行）

finish：结束当前函数，返回到函数调用点

continue( c )：继续运行

list(l)：查看源代码（list-n,从第n行开始查看代码。list+ 函数名：查看具体函数）

info(i)：查看函数内部局部变量的数值

print( p )：打印值及地址

display：追踪查看具体变量值

undisplay：取消追踪观察变量

info breakpoints：查看当前设置的所有断点

watch：被设置观察点的变量发生修改时，打印显示

i watch：显示观察点

x：查看内存x/20xw 显示20个单元，16进制，4字节每单元

break+num(b)：在第num行设置断点

info breakpoints：查看当前设置的所有断点

delete breakpoints num(d)：删除第num个断点

enable breakpoints：启用断点

disable breakpoints：禁用断点

backtrace(bt) ：查看函数的调用的栈帧和层级关系       frame(f)：切换函数的栈帧       info(i)：查看函数内部局部变量的数值       finish：结束当前函数，返回到函数调用点

set：设置变量的值       display：追踪查看具体变量值       undisplay：取消追踪观察变量       watch：被设置观察点的变量发生修改时，打印显示       i watch：显示观察点

3.3 应用程序交互监控

分章节讲述对程序的系统调用和交互的监控

3.3.1 ltrace

ltrace是一个简单地运行指定命令的程序，直到它退出。它拦截并记录动态库调用其由执行的过程调用由该过程接收。它还可以拦截和打印由程序执行的系统调用。

-a, --align column

对齐特定列中的返回值（默认列是屏幕宽度的5/8）。

 -A maxelts

之前要打印的最大数组元素数用省略号（“…”）抑制其余部分。这也是限制递归结构扩展的数量。

 -b, --no-signals

禁止打印被跟踪者接收的信号过程

-c计算每次库调用的时间和调用，并报告程序退出时的摘要。

-C, --demangle

将低级符号名称解码（解映射）为用户级名称。除了删除任何初始下划线前缀这使得C++函数名可读。

-D, --debug mask

显示ltrace本身的调试输出。mask是一个数字描述应显示哪些调试消息。使用选项-Dh查看可以使用的内容，但请注意目前唯一可靠的调试掩码是77，它显示所有调试消息。

-e filter

修改库调用的限定表达式（即通过PLT插槽进行的调用从主二进制文件到库或库间文件的调用调用）进行跟踪。用法示例和语法描述出现在以下过滤器规格和过滤器部分表达。如果在命令行，匹配其中任何一个的库调用是跟踪。如果未给出-e，则假定@MAIN为默认值。

-f跟踪当前创建的子进程

由于分叉（2）或克隆（2）而跟踪的进程系统调用。新流程将立即附加。

-F、 --config pathlist

包含以冒号分隔的路径列表。如果路径指目录，当搜索原型库（请参见原型库发现）。如果它指的是文件，该文件将隐式导入所有加载的原型库。

-i在库运行时打印指令指针调用

-l、--library library\_pattern

仅显示对库实现的函数的调用匹配library\_patter。这就像您指定的指定库中实现的每个符号一个-e模式。可以使用多种库模式使用此选项的多个实例指定。用法librarypattern的示例和语法描述出现在以下过滤器规格和过滤器部分表达。请注意，虽然此选项选择可能指向选定的库保证由于以下原因，电话不会被转接到其他地方e、 g.LD\_PRELOAD或简单的依赖排序。如果你希望确保给定库中的符号实际调用时，请改用-x@library\_path。

-L当没有给定-e选项时，不要假定默认值@MAIN的动作。实际上，这意味着图书馆调用将不会被跟踪。

-n、 --indent nr

按每个调用级别的nr个空格缩进跟踪输出嵌套。使用此选项可使程序流动可视化易于理解。这种缩进也是无用的永不返回的函数，例如的服务函数在C++运行时引发异常。

-o、--output filename

将跟踪输出写入文件名，而不是写入标准错误。

-p pid附加到具有进程ID pid的进程并开始跟踪。此选项可以与传递要执行的命令。可以连接到多个通过传递多个选项-p来处理。

-r打印跟踪的每一行的相对时间戳，这记录了连续行。

-s strsize指定要打印的最大字符串大小（默认值为32).

-S显示系统调用和库调用

-t在跟踪的每行前面加上一天的时间。

-tt如果给出两次，打印的时间将包括微秒。

-ttt如果给三次，打印的时间将增加

3.3.2 strace

strace命令是一个集诊断、调试、统计于一体的工具,常用来跟踪进程执行时的系统调用和所接收的信号，我们可以用它来监控用户空间进程和内核的交互。如对应用程序的系统调用、信号传递与进程状态变更等进行跟踪与分析，以达到解决问题的目的。       strace常用来跟踪进程执行时的系统调用和接收的信号。 在Linux中，进程不能直接访问硬件设备，当进程需要访问硬件设备(比如，读取磁盘文件，接收网络数据等等)时，必须由用户态模式切换至内核态模式，通过系统调用访问硬件设备。strace可以跟踪到一个进程产生的系统调用，包括参数，返回值，执行消耗的时间。

-c 统计每一系统调用的所执行的时间,次数和出错的次数等；

-d 输出strace关于标准错误的调试信息

-f 跟踪由fork调用所产生的子进程

-ff 如果提供-o filename,则所有进程的跟踪结果输出到相应的filename.pid中,pid是各进程的进程号

-F 尝试跟踪vfork调用.在-f时,vfork不被跟踪

-h 输出简要的帮助信息

-i 输出系统调用的入口指针

-q 禁止输出关于脱离的消息

-r 打印出相对时间关于每一个系统调用

-t 在输出中的每一行前加上时间信息

-tt 在输出中的每一行前加上时间信息,微秒级

-ttt 微秒级输出,以秒了表示时间

-T 显示每一调用所耗的时间

-v 输出所有的系统调用，一些调用关于环境变量,状态,输入输出等调用由于使用频繁,默认不输出

-V 输出strace的版本信息

-x 以十六进制形式输出非标准字符串

-xx 所有字符串以十六进制形式输出

-a column设置返回值的输出位置.默认 为40

-e expr指定一个表达式,用来控制如何跟踪.格式如下:

[qualifier=][!]value1[,value2]...

qualifier只能是 trace,abbrev,verbose,raw,signal,read,write其中之一，value是用来限定的符号或数字，默认的 qualifier是 trace，感叹号是否定符号，例如:-eopen等价于 -e trace=open,表示只跟踪open调用，而-etrace!=open表示跟踪除了open以外的其他调用.有两个特殊的符号 all 和 none.注意有些shell使用!来执行历史记录里的命令,所以要使用\\

-e trace=set只跟踪指定的系统调用.例如:

-e trace=open,close,rean,write表示只跟踪这四个系统调用.默认的为set=all

-e trace=file只跟踪有关文件操作的系统调用

-e trace=process只跟踪有关进程控制的系统调用

-e trace=network跟踪与网络有关的所有系统调用

-e strace=signal跟踪所有与系统信号有关的系统调用

-e trace=ipc跟踪所有与进程通讯有关的系统调用

-e abbrev=set设定 strace输出的系统调用的结果集

-v 等与 abbrev=none，默认为abbrev=all

-e raw=set将指 定的系统调用的参数以十六进制显示

-e signal=set指定跟踪的系统信号.默认为all，如signal=!SIGIO(或者signal=!io),表示不跟踪SIGIO信号

-e read=set输出从指定文件中读出 的数据.例如:

-e read=3,5-e write=set输出写入到指定文件中的数据

-o filename将strace的输出写入文件filename-p pid跟踪指定的进程pid

-s strsize指定输出的字符串的最大长度.默认为32.文件名一直全部输出

-u username以username 的UID和GID执行被跟踪的命令

4 软件发布

4.1 源码包发布

分章节讲述标准源码tar.gz包制作步骤和发布

4.1.1 安装autoconf、automake软件包

[root@localhost Hello\_Aotumake]#yum install autoconf automake aclocal

4.1.2 创建测试程序

在/home/rpmuser目录下创建Hello\_Automake目录，并在目录下创建Hello\_Automake.c文件。文件内容如下：

[rpmuser@localhost Hello\_Aotumake]# cat Hello\_Automake.c

#include <stdio.h>

int main(){

printf("Hello automake");

return 0;

}

4.1.3 生成configure.ac文件

在/home/rpmuser/Hello\_Automake/目录下执行autoscan命令

[rpmuser@localhost Hello\_Aotumake]$ autoscan

查看生成的文件

[rpmuser@localhost Hello\_Aotumake]$ ls

autoscan.log configure.scan Hello\_Automake.c

将configure.scan文件改名为configure.ac，修改内容如下：

[rpmuser@localhost Hello\_Aotumake]$ cat configure.ac

# -\*- Autoconf -\*-

# Process this file with autoconf to produce a configure script.

AC\_PREREQ([2.69])

AC\_INIT([Hello\_Automake], [1.0], [lvcongqing@uniontech.com])

AC\_CONFIG\_SRCDIR([Hello\_Automake.c])

AC\_CONFIG\_HEADERS([config.h])

AM\_INIT\_AUTOMAKE(Hello\_Automake,1.0)

AC\_PROG\_CC

AC\_OUTPUT([Makefile])

4.1.4 生成aclocal.m4文件

执行aclocal命令，将configure.ac文件所需要的宏集中定义到文件 aclocal.m4中

[rpmuser@localhost Hello\_Aotumake]$ aclocal

[rpmuser@localhost Hello\_Aotumake]$ ls

aclocal.m4 AUTHORS autom4te.cache autoscan.log ChangeLog configure.ac INSTALL Makefile.am NEWS README Hello\_Automake.c

4.1.5 生成configure文件

执行autoconf命令，将configure.ac中的宏展开，生成configure脚本

[rpmuser@localhost Hello\_Aotumake]$ autoconf

[rpmuser@localhost Hello\_Aotumake]$ ls

aclocal.m4 autom4te.cache ChangeLog configure.ac Makefile.am README

AUTHORS autoscan.log configure INSTALL NEWS Hello\_Automake.c

4.1.6 生成config.h.in文件

执行autoheader命令，创建C宏定义模板文件config.h.in

[rpmuser@localhost Hello\_Aotumake]$ autoheader

[rpmuser@localhost Hello\_Aotumake]$ ls

aclocal.m4 autom4te.cache ChangeLog configure INSTALL NEWS Hello\_Automake.c

AUTHORS autoscan.log config.h.in configure.ac Makefile.am README

4.1.7 生成Makefile文件

创建Makefile.am文件，并添加内容如下，Automake工具会根据 configure.in 中的参量把 Makefile.am 转换成 Makefile.in 文件。最终通过Makefile.in生成Makefile文件

[rpmuser@localhost Hello\_Aotumake]$ cat Makefile.am

AUTOMAKE\_OPTIONS=gnu

bin\_PROGRAMS=Hello\_Automake

main\_SOURCES=Hello\_Automake.c

执行automake命令，生成 Makefile.in 文件。使用选项 "--add-missing" 可以让 Automake 自动添加一些必需的脚本文件。如果发现一些文件不存在，可以通过手工 touch命令创建。这里需要手工创建NEWS、README、ChangeLog、AUTHORS四个文件。

[rpmuser@localhost Hello\_Aotumake]$ automake --add-missing

[rpmuser@localhost Hello\_Aotumake]$ ls

aclocal.m4 autom4te.cache ChangeLog configure COPYING INSTALL Makefile.am missing README

AUTHORS autoscan.log config.h.in configure.ac depcomp install-sh Makefile.in NEWS Hello\_Automake.c

4.1.8 预编译

执行configure命令进行预编译，生成Makefile，为下一步编译做准备。

[rpmuser@localhost Hello\_Aotumake]$ ./configure

4.1.9 执行编译

执行make命令进行编译，生成可执行文件Hello\_Automake

[rpmuser@localhost Hello\_Aotumake]$ make

清理编译产生的过程文件和可执行文件Hello\_Automake

[rpmuser@localhost Hello\_Aotumake]$ make clean

[rpmuser@localhost Hello\_Aotumake]$ test -z "Hello\_Automake" || rm -f Hello\_Automake

[rpmuser@localhost Hello\_Aotumake]$ rm -f \*.o

4.1.10 生成发布文件

执行make dist命令，生成Hello\_Automake程序的发布压缩文件Hello\_Automake-1.0.tar.gz

[rpmuser@localhost Hello\_Aotumake]$ make dist

[rpmuser@localhost Hello\_Aotumake]$ ls

aclocal.m4 ChangeLog config.status COPYING INSTALL Makefile.in stamp-h1

AUTHORS config.h configure depcomp install-sh missing

autom4te.cache config.h.in configure.ac Hello\_Automake-1.0.tar.gz Makefile NEWS

autoscan.log config.log configure.scan Hello\_Automake.c Makefile.am README

4.2 rpm包发布

4.2.1 创建测试程序

将上一章节生成的Hello\_Automake程序的发布压缩文件Hello\_Automake-1.0.tar.gz复制到/home/rpmuser/rpmbuild/SOURCES目录下面

[rpmuser@localhost Hello\_Aotumake]$ cp Hello\_Automake-1.0.tar.gz /home/rpmuser/rpmbuild/SOURCES/

4.2.2 编写SPEC文件

在/home/rpmuser/rpmbuild/目录下面创建Hello\_Automake.spec文件，并添加内容如下：

[rpmuser@localhost SPECS]$ vim Hello\_Automake.spec

Name:Hello\_Automake

Version: 1.0

Release: 1%{?dist}

Summary: Hello automake test!

Group: uniontech.com

License: GPL

URL: www.uniontech.com

Source0: Hello\_Automake-1.0.tar.gz

#BuildRequires:

#Requires:

%description

%prep

%setup -q

%build

%configure

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%defattr(-,root,root)

%doc ChangeLog INSTALL README AUTHORS COPYING

%attr(0755,root,root) %{\_bindir}/\*

%changelog

4.2.3 生成rpm包

执行rpmbuild命令，制作rpm包。

[rpmuser@localhost rpmbuild]$ rpmbuild -ba SPEC/Hello\_Automake.spec

进入到/home/rpmuser/rpmbuild/RPMS/x86\_64目录下面查看生成的rpm文件。

[rpmuser@localhost x86\_64]$ rpm -pql Hello\_Automake-1.0-1.el7.x86\_64.rpm

/usr/bin/Hello\_Automake

/usr/share/doc/Hello\_Automake-1.0

/usr/share/doc/Hello\_Automake-1.0/AUTHORS

/usr/share/doc/Hello\_Automake-1.0/COPYING

/usr/share/doc/Hello\_Automake-1.0/ChangeLog

/usr/share/doc/Hello\_Automake-1.0/INSTALL

/usr/share/doc/Hello\_Automake-1.0/README

进入到/home/rpmuser/rpmbuild/SRPMS/目录下面查看生成的src rpm文件。

[rpmuser@localhost SRPMS]$ rpm -pql Hello\_Automake-1.0-1.el7.src.rpm

Hello\_Automake-1.0.tar.gz

Hello\_Automake.spec

©统信软件技术有限公司。访问者可将本网站提供的内容或服务用于个人学习、研究或欣赏，以及其他非商业性或非盈利性用途，但同时应遵守著作权法及其他相关法律的规定，不得侵犯本网站及相关权利人的合法权利。除此以外，将本网站任何内容或服务进行转载，须备注：**该文档出自【faq.uniontech.com】统信软件知识分享平台**。否则统信软件将追究相关版权责任。