Software aus dem Sourcecode Installieren:

1. Welche Vor und Nachteile hat hat Software die aus dem Sourcecode Installiert wird gegenüber den Vorgefertigten varianten (rpm, deb, ...)
2. Welche Voraussetzungen müssen erfüllt sein.
3. Wie geht man bei der Installation vor?

zu 1.) Vorteile:
- Kompatibel zu (fast) allen Distributionen und Architekturen
- Besser Anpassbar
- Programm spezifische Optionen können Aktiviert oder Deaktiviert werden

Nachteile:
- Keine Version und Abhängigkeitsprüfung
- erfordert mehr Wissen
- Dauert u.U sehr lange

zu 2.)
- Compiler
* - gcc
- c++
- linux-headers
- glibc
* - make
* - automake
- Sourcecode des zu installierenden Programms

zu 3.) download des Sourcecodes in der Form:

programm.tar.gz
programm.tgz
programm.tar.bz2
programm.tbz
programm.tbz2

Kopieren des Sourcecodes nach /usr/local/src:

cp programm.tar.gz /usr/local/src

Entpacken des Sourcecodes:

unzippen: ***gunzip programm.tar.gz***
bunzip2 programm.tar.bz2
auspacken: ***tar -xf programm.tar***
zusammen: ***tar -xzf programm.tar.gz***
tar -xjf programm.tar.bz2

Nun im Source Ordner "./configure" ausführen

./configure --prefix=/usr/local/samba --with-ldap --enable-cups

Dies baut nun ein MakeFile, Samba soll Später nach "/usr/local/samba" installiert werden, Samba soll mit CUPS Funktionalität Übersetzt werden und mit dem Programm LDAP zusammenarbeiten.

Wenn Das ./configure erfolgreich war kann Samba mit "make" Übersetzt werden

make

Und dann mit "make install" installiert werden.

Andere "make Ziele":

make clean -> Bereits Übersetzte Programmteile werden entfernt
make mrproper -> Alles wird entfernt (auch das MakeFile)
make uninstall -> Das Programm wird Deinstalliert.

Verwaltung von Bibliotheken:

Unter Linux gibt es sehr viele Bibliotheken. Es gibt ein Programm welches jedes zu Startende Programm untersucht, welche Bibliotheken es benötigt, und diese Bibliotheken dann vor dem eigentlichem Programm Läd. Diese Programm heißt meistens „***ld-linux.so***" Da es nun sehr lange dauern würde auf der Gesammte Festplatte nach den erforderlichen Bibliotheken zu suchen, wird nur in 2 Verzeichnissen nach Bibliotheken gesucht: ***/lib*** und ***/usr/lib***, jedoch nicht in deren Unterverzeichnissen. Sollte die gesuchte Bibliothek dort nicht gefunden werden wird eine Datenbank befragt. Die ***/etc/ld.so.cache*** Diese Datenbank wird durch den Befehl ***ldconfig*** Aktualisiert. Der Befehl ***ldconfig*** durchsucht alle in der ***/etc/ld.so.conf*** angegebenen Verzeichnisse nach Bibliotheken und trägt diese in die Datenbank ein, auch hier werden Unterverzeichnisse nich beachtet. Mit dem Befehl ***ldconfig -p*** werden alle dem System bekannten Bibliotheken angezeigt.

Es gibt noch eine weitere Methode dem System Bibliotheken bekannt zu machen. Diese sollte jedoch nur Temporär oder zu Testzweken genutzt werden. Will ich z.B.: das Bibliotheken im Verzeichnis /usr/local/test/lib dem System bekannt seinen sollen, kann ich die Variabel ***LD_LIBRARY_PATH*** dazu verwenden.
export LD_LIBRARY_PATH=/usr/local/test/lib
In den Verzeichnissen die mit dieser Variabel dem System bekannt gemacht wurden, wird immer zu erst nach einer Bibliothek gesucht. Mit einer Ausnahme, Programme mit gesetztem SUID und SGID Bit, beachten diese Variabel nicht.

Um heraus zu finden, welch Bibliotheken in Programm benötigt, wird der Befehl ldd verwendet. z.B:

ldd /bin/ls
gibt folgende Ausgabe:

```
linux-gate.so.1 =>  (0xffffe000)
        librt.so.1 => /lib/tls/i686/cmov/librt.so.1 (0xb7f01000)
        libacl.so.1 => /lib/libacl.so.1 (0xb7efb000)
        libselinux.so.1 => /lib/libselinux.so.1 (0xb7ee5000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7db4000)
        libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0
(0xb7da2000)
        /lib/ld-linux.so.2 (0xb7f1d000)
        libattr.so.1 => /lib/libattr.so.1 (0xb7d9e000)
        libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7d9a000)
        libsepol.so.1 => /lib/libsepol.so.1 (0xb7d59000)
```

Die Debian Paketverwaltung:

Namen eines Debian-Paketes:

```
  <Programmname>_<Version>_<arch>.deb
```

Die .deb Datei ist ein AR-Archiv (Modifiziertes Tar-Archiv)
Darin befinden sich 3 Dateien:

```
      Debian-binary     →  Versionsnummer des Paketformates
      control.tar.gz    →  Metadaten des Programms
      data.tar.gz       →  Das eingentliche Programm
```

Informationen über Bereitz installierte Programme:

Im Ordner /var/lib/dpkg/

```
  available       -> Alle Packete die man Installieren könnte
  status          -> Installationsstatus aller Installierten Pakete
  ./info/*        -> Informationen über Installierte Pakete
    paket.list          -> Liste aller Dateien die zum Paket gehören
    paket.md5sums -> MD5-Summen aller Dateien aus dem Paket
    paket.shlibs  -> Vom Paket bereit gestellte Bibliotheken
    paket.preinst -> Shell-Script, welches vor der Installation
                     ausgeführt wird.
    paket.postinst      -> Shell-Script, welches nach der Installation
                           ausgeführt wird.
    paket.prerm         -> Shell-Script, welches vor der Deinstallation
                           ausgeführt wird.
    paket.postrm  -> Shell-Script, welches nach der Deinstallation
                           ausgeführt wird.
    paket.conffiles     -> Eine Liste der Konfigurationsdateien des Paketes
```

Das Programm dpkg:

Installieren:

```
  dpkg -i <Paket>.deb          →  Installiert ein Paket
  dpkg -i --force-depends      →  Installiert alle Pakete auch wenn
                                  abhängigkeiten nicht erfüllt sind
  dpkg -i --ignore-depends=paket.dep →  Ignoriert abhängigkeiten für das Paket
```

dpkg -G | --refuse-downgrade → Wenn schon ein neueres Paket vorhanden ist, wird kein älteres Installiert.

Deinstallieren:

dpkg -r | --remove → Deinstalliert ein Paket, aber die Konfigurationsdateien bleiben erhalten
dpkg -P | --purge → Deinstalliert ein Paket und die Konfigurationsdateien
dpkg -B | --auto-deconfigure → Das Paket wird entfernt und alle Pakete die es benötigen werden Dekonfiguriert.

Informationen:

dpkg -l | --list → Zeigt alle Installierten Pakete
dpkg -S | --search /bin/ls → Zeigt an zu welchem Paket /bin/ls gehört
dpkg -L | --listfiles paket → Zeigt alle Dateien die zu diesem Paket gehören
dpkg -s | --status paket → Zeigt den Installationsstatus des Paketes und die Informationen zu dem Paket

Available Datenbank Aktualisieren:

dpkg --update-avail Paketdatei → Die alte DB wird durch die neue ersetzt
dpkg --merge-avail Paketdatei → Die neue und die alte DB werden verbunden

Andere Befehle:

dpkg-reconfigure paket → Bereitz Konfigurierte Programme, neu Konfigurieren.
dselect → Menügesteuertes Frontend für dpkg

Die APT-Programme:

/etc/apt/apt.conf → Konfigurationsdatei für ältere Apt-Programme
/etc/apt/apt.conf.d/* → Konfigurationsdateien für neuere Apt-Prog.
/etc/apt/sources.list → Hier stehen die Quellen für die Available Datenbank (CD-Rom, HTTP, www, FTP ...)

apt-get update → Die Available DB wird Aktualisiert
apt-get upgrade → Aktualisiert alle Installierten Pakete
apt-get install mplayer → Installiert den MPlayer und alle Abhängigkeiten
apt-get source mplayer → Installiert dem MPlayer als Sourcecode Vers.
apt-get remove mplayer → Deinstalliert den MPlayer
apt-get install mplayer- → Deinstalliert den MPlayer

apt-get install paket1 paket2- paket3 paket4- → Hier werden nun die Pakete „paket1 und paket3“ installiert und „paket2 und paket4“ deinstalliert.

apt-get install mplayer* → Versucht alle Pakete, die "mplayer" enthalten zu installieren.

Alien:

alien --to-deb paket.rpm → Aus dem RPM-Paket wird ein DEB-Paket
--to-rpm paket.deb → Aus dem DEB-Paket wird ein RPM-Paket
--to-tgz paket.rpm → Aus dem RPM-Paket wird ein Slackware-Paket
--to-pkg paket.rpm → Aus dem RPM-Paket wird ein Solaris-Paket
--to-slp paket.rpm → Aus dem RPM-Paket wird ein Stampede-Paket

RPM-Pakete:

Namen von RPM-Paketen:

  <Programmname>-<Version>.<arch>.rpm

Die RPM-Datenbank:

  /var/lib/rpm/* -> Die RPM-Datenbank
  /usr/lib/rpm/* -> Hilfsprogamme für RPM
  /etc/rpmrc | ***~/.rpmrc*** | ***/usr/lib/rpmrc*** | mit ***–rcfile*** angegebene Datei
                    Konfigurationsdatei

RPM-Installieren:

Vor der Installation:

rpm -qip paket.rpm → Informationen über das (nicht Installierte) Paket ausgeben.
rpm --checksig paket → Überprüft die MD5 (SHA1, SHA2, DSA, ...) Summe, mit der ich feststellen ob das Paket beschädigt ist. Desweiteren wird die PGP oder GPG überprüft, mit der ich Feststellen kann ob das Paket wirklich von da kommt, wo es herkommen soll (z.B.: von SuSE)

Installieren: (-i)

rpm -i | --install Paket.rpm -> Installiert das Paket, wenn alle Abhängigkeiten erfüllt sind.

- ***--test*** → Testen ob sich das Prog. Installieren ließe
- ***--nodeps*** → Ignoriert Abhängigkeiten
- ***--noscripts*** -> Pre/Post Installscripte werden nicht Ausgeführt.
- ***--excludedocs*** → Dokumentationen und Man-Pages werden nicht mit installiert
- ***--replacepkgs*** → Installiert das Paket uach wenn schon teile vorhanden sind
- ***--replacefiles*** → Installier das Paket, auch wenn dabei Teile die von anderen Paketen stammen überschrieben werden.
- ***--oldpackage*** → Mit dieser Option kann ein älteres Paket, als das bereits vorhandene, Installiert werden
- ***--force*** → entspricht: ***--replacepkgs*** ***--replacefiles*** und ***--oldpackage***

Deinstallieren (-e)

rpm -e paket<-version> → Deinstalliert das Paket

- ***--test*** → Testen ob sich das Paket Deinstallieren läßt
- ***--nodeps*** → Ignoriert Abhängigkeiten
- ***--noscripts*** → Pre/Post Deinstallscripte werden nicht Ausgeführt.
- ***--allmatches*** → Alle Versionen dieses Paketes werden Deinstalliert
- ***--repackage*** → Baut aus den zu Deinstallierenden Dateien wieder ein RPM-Paket, diese befindet sich dann unter: /var/spool/repackage/

Updaten von RPM-Paketen:

rpm -U | --upgrade paket.rpm	→	Wenn das Paket in einer älteren Version vorhanden ist, wird es Aktualisiert, wenn nicht wird es Installiert
rpm -F | --freshen paket.rpm	→	Nur wenn eine Ältere Version des Paketes vorhanden ist, wird es Aktualisiert, wenn nicht wird es Ignoriert.

--> Es können die selben Optionen wie beim Installieren Verwendet werden.

Informationen über Pakete erfragen: (-q)

Informationen über bereits installierte Pakete immer ohne die Option "-p" und für noch nicht Installierte Pakete immer mit "-p"

rpm -qi paket	→	Informationen über das Paket
-ql paket	→	Zeigt eine Liste aller Dateien die zu diesem Paket gehören.
-qlv paket	→	Ähnlich wie -ql, entspricht der Ausgabe von ls -l
-qd paket	→	Zeigt alle "doc" Dateien aus dem Paket
-qc pacet	→	Zeigt alle "config" Dateien aus dem Paket
-qf /bin/ls	→	Zeigt zu welchem Paket "/bin/ls" gehört
-qR paket	→	Zeigt an welche Bibliotheken von diesem Paket benötigt werden
-q --whatrequires coreutils	→	Zeigt an welche Pakete das Paket "coreutils" benötigen.

-i = --install | -q = --query | -e = --erase | -i = --info | -l = --list
-c = --configfiles | -d = --docfiles | -v = --verbose | -R = --requires
-f = --file | -p = --package | -V = --verify

Installierte Pakete auf Integrität Prüfen:

rpm -V coreutils	→	Prüft die Integrität aller Dateien aus dem Paket coreutils
rpm -aV	→	Prüft die Integrität aller Dateien aus allen Paketen.

Ausgabe könnte folgendermaßen aussehen:

```
S.5....T  c /etc/ssh/sshd_config
```

S = Die Größe der Datei hat sich verändert
M = Zugriffsrechte stimmen nicht mehr
5 = Die MD5-Summe stimmt nicht mehr
D = Major/Minor Nummer einer Gerätedatei stimmen nicht mehr
L = Sollte eine Datei sein, ist aber ein Link
U = Eigentümer stimmt nicht mehr
G = Gruppenzugehörigkeit stimmt nicht mehr
T = die mtime stimmt nicht mehr

d = Es handelt sich um eine "doc" Datei
c = Es handelt sich um eine "config" Datei