# Essential Skills. Lab 2

Ali Abdulmadzhidov

09 September 2016

## 1 Debian packagins system

### 1.1 How does it work?

Debian package system has various different level tools for user. The base of package system is dpkg - Debian package manager. In spite of man page says that the description of steps in installing or removing process is inadequate, i'll try.

#### *.deb - Debian package

Debian package contains three sections:

**Global header** Contains version of debian package. Current version is 2.0

**Control section** Meta information about package. Conffiles, preinst, postinst, prerm, postrm scripts, list of dependecies, md5 hashsumm, brief description.

**Data section** Includes installable files

#### Installation

dpkg -i

1 Extract of control files of the package

2 If another version of package is already installed, execute prerm script of the outdated package

3 If package has preinst script, execute it.

4 Run postrm script of old package

5 Unpack new, and at the same time backup old files.

6 Configure package

#### Configuration

dpkg -configure

1 Unpack conffiles, and backup old conffiles.

2 Run postinit script

#### Removing

dpkg -remove
Removes an installed package, but lefts conffiles

1 Run prerm script

2 Remove isntalled files

3 Run posterm script

#### Purge

dpkg -purge
Purges (removes) an installed or removed package with conffiles.

1 Remove package

2 Run posterm script

### 1.2 Tech requirments

1. build-essential
2. dev scripts
3. Root rights to install something

### 1.3 How doest it deals with dependecies?

Debian package dependecies list is stored Control Section of .deb package. By default dpkg doesn't download all dependecies, but after installation we can call apt or another tool to install them.

```
apt -f install
```

This command will find all broken dependecies and install them for all packages

### 1.4 Does it use the GNU build tools? How?

Debian Package System doesn't use autotools for installation or other common actions with it's packets. But autotools are needed to build .deb packets from sources.

## 2 OpenSUSE packaging system

### 2.1 How does it work?

OpenSUSE package is a bit simillar to debian one's. But it uses rpm packages, that are considered to be standartized for linux by foundation.

***.rpm - RPM Package Manager/Red Hat Package Manager**

Originally was created for Red Hat linux, but was added to LBS standat. By default has name

```
<name>-<version>-<release>.<architecture>.rpm
```

A package also has metadata, such as:

1. summary,
2. description,
3. list of files contained in the package,
4. version of the software it contains and release number of the package,
5. information about where and whern this package was built and it's author,
6. what architecture it has been built for,
7. checksums of the files contained in the package,
8. license of the software it contains,
9. dependecies

#### 2.1.1 Usage

**Installation** `zypper install <package>.rpm/<package_name_in_repo>`

**Remove** `zypper remove <package>.rpm/<package_name_in_repo>`

**Update** `zypper update <package>.rpm/<package_name_in_repo>`

**Search** `zypper searche <key>`

**Build from srpm** `zypper source-install <package>.srpm/<package_name_in_repo>`

**Info** `zypper info <package>.rpm/<package_name_in_repo>`

## 2.2 How doest it deals with dependecies?

Zypper package manager loads all dependecies by list that from package. RPM package manager leaves this stuff to user. He should install all deps manually.

## 2.3 Does it use the GNU build tools? How?

OpenSUSE package managers doesn't need autotools to install packages, cause they downloadas already builded, but you need autotools to rebuild rpm from sources or srpm.

## 3 Assignment 2. Installing pacman clone from sources

### 3.1 Building for our platform

```
wget https://sourceforge.net/projects/myman/files/latest/download # Download sources
tar -xvf download # Extraction from archive
cd myman-wip-2009-10-30 # Go to src dir
./configure # Use autotools to configurate for our platform
make # Try to build. Find missing dependicies
sudo apt install libncurses5-dev # Install them
sudo apt install groff #
make # Second try
make install # Installation
```

### 3.2 Building for i386

```
wget https://sourceforge.net/projects/myman/files/latest/download # Download sources
tar -xvf download # Extraction from archive
cd myman-wip-2009-10-30 # Go to src dir
./configure CC='gcc -m32' CXX='g++ -m32'# Use autotools to configurate for i386 platform
make # Try to build. Find missing dependicies
sudo apt install libncurses5-dev:i386 # Install them
sudo apt install groff:i386 #
make # Second try
make install # Installation
file myman # checking result. We should see that this app is compiled for
myman: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter
```

Autotools containts

1. Autoconf that reads configure.in and make ./configure script
2. Automake that makes Makefile.in from Makefile.am, that in their order reads by ./configure and becomes Make file for make
3. Libtool that manages making static or dinamic libs in Unix.

Attached screenshots of working game and file type.