ОС UNIX. Практика 2
Оформление проектов и инструменты сборки

Михаил Пожидаев

25 сентября 2025 г.

Уровни инструментария разработки

1. Системы управления пакетами (rpm и dpkg).
2. Системы сборки проектов (make, autoconf, cmake и scons).
3. Компиляторы и линкеры (gcc, g++ и пр.).

Стандартные утилиты gcc

1. `gcc` — компилятор и линкер для языка C.
2. `g++` — компилятор и линкер для языка C++.
3. `gdb` — отладчик для проектов gcc.
4. `ldd` — инспектор динамически подключаемых зависимостей.

Использование gcc

Пример сборки приложения:

```
gcc -Wall -pedantic -O2 -omyprog myprog.c -lpthread
```

Пример сборки разделяемой библиотеки:

```
gcc -Wall -pedantic -fpic -O2 -shared myprog.c
```

Библиотеки в GNU/Linux

1. Статические, имена файлов обычно имеют вид `libfoobar.a`.
2. Динамические, имена файлов имеют вид `libfoobar.so`.

Необходимо быть аккуратным с числами версий разделяемых библиотек. Первое число версии указывает на сохранение совместимости, его увеличение обозначает, что библиотека потеряла совместимость с предыдущими версиями.

Дополнительные инструменты разработки и отладки

1. `doxygen` — система автоматической генерации документации.
2. `strace` — трекер системных вызовов.
3. `valgrind` — трекер утечек памяти.

Пример вывода strace

```
$ echo 'university' > proba
      $ strace cat proba
...
open("proba", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0664, st_size=11, ...}) = 0
fadvise64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, ...) = 0x7fb7633c6000
read(3, "university\n", 131072) = 11
write(1, "university\n", 11university) = 11
read(3, "", 131072) = 0
```

Пример Makefile

```
all: myprog

myprog: myprog.c
gcc -Wall -pedantic -O2 -omyprog myprog.c
```

Кроссплатформенные сборщики

1. *cmake* — кроссплатформенный сборщик с собственным синтаксисом файлов.
2. *scons* — кроссплатформенный сборщик на основе Python.

Пример configure.ac для autoconf

```
AC_INIT(voiceman, 1.5.0.3, msp@altlinux.org)
AC_CONFIG_MACRO_DIR([m4])
AM_INIT_AUTOMAKE([-Wall foreign])
if test "x$CFLAGS" == 'x'; then
   CFLAGS='-O2'
fi
if test "x$CXXFLAGS" == 'x'; then
   CXXFLAGS='-O2'
fi
AC_PROG_CC
AC_PROG_CXX
AC_PROG_RANLIB
....
AC_OUTPUT(voiceman.conf)
AC_CHECK_LIB(ao, ao_initialize, [], [...])
```

Пример файла для automake

```
AM_CXXFLAGS = $(VOICEMAN_CXXFLAGS) $(VOICEMAN_INCLUDES)

bin_PROGRAMS = voiceman-trim

voiceman_trim_SOURCES = trim.cpp
```

Структура spec-файла для RPM

1. Заголовок основного пакета с текстовыми полями и зависимостями.
2. Заголовки подпакетов.
3. Расширенные описания.
4. Инструкции для сборки пакета:
   - `%setup` — подготовка исходников;
   - `%build` — непосредственно сборка;
   - `%install` — раскладка файлов после сборки.
5. Списки файлов для всех подпакетов.
6. История обновлений.

Заголовок spec-файла для RPM

```
Name: voiceman
Version: 1.5.0.2
Release: alt2
License: %gpl3plus
URL: http://www.marigostra.ru/projects/voiceman/
Summary: Universal server for processing speech output
Group: Sound
Source: %name-%version.tar.gz
BuildRequires: rpm-build-licenses gcc-c++ libao-devel
```

Команды сборки RPM

```
%prep
%setup -q
%build
%autoreconf
%configure default_socket=/var/run/voiceman.socket
%make_build
%install
make DESTDIR=%buildroot install
%__rm -f %buildroot%_sysconfdir/%name.conf
```

Умолчательный файл rules для dpkg

```
#!/usr/bin/make -f
# -*- makefile -*-
# Uncomment this to turn on verbose mode.
#export DH_VERBOSE=1
# This has to be exported to make some magic below work.
export DH_OPTIONS
%:
dh $@  --with autotools-dev
```

Спасибо за внимание!

Всё о курсе: https://marigostra.ru/materials/unix.html
Канал в Телеграм: https://t.me/MarigostraRu