Programovanie v oeračných systémoch

12 - Build, Tools, Kernel modules

Jozef Šiška

Department of Applied Informatics
Comenius University in Bratislava

2016/2017

Build systems

- Script
- Makefile
- The nine^Wthree rings^Wcommands to rule^Winstall them all
  - portability, customization
- autotools / autoconf
- CMake, SCons, waf, . . .

Build systems

- Script
- Makefile
- The nine^Wthree rings^Wcommands to rule^Winstall them all
  - portability, customization
- autotools / autoconf
- CMake, SCons, waf, . . .

```
p: a.o b.o
%.o: %.c
	$(CC) -c $(CFLAGS) $(CPPFLAGS) $< -o $@
clean:
	rm -f p *.o
```

Build systems

- Script
- Makefile
- The nine^Wthree rings^Wcommands to rule^Winstall them all
  - portability, customization
- autotools / autoconf
- CMake, SCons, waf, . . .

```
./configure
make
make install
```

Build systems

- Script
- Makefile
- The nine^Wthree rings^Wcommands to rule^Winstall them all
  - portability, customization
- autotools / autoconf
- CMake, SCons, waf, . . .

```
CC='ccache gcc' ./configure --prefix=${HOME}/.local/
make -j5
make install
```

Build systems

- Script
- Makefile
- The nine^Wthree rings^Wcommands to rule^Winstall them all
  - portability, customization
- autotools / autoconf
- CMake, SCons, waf, . . .

```
AC_INIT([package], [version])
AM_INIT_AUTOMAKE([foreign subdir-objects])
AC_CONFIG_SRCDIR([configure.ac])
AC_PROG_CC      # or AC_PROG_CXX
AC_CONFIG_FILES([Makefile])
PKG_CHECK_MODULES([cairo], [cairo])
PKG_CHECK_MODULES([fontconfig], [fontconfig])
AC_OUTPUT
```

Build systems

- Script
- Makefile
- The nine^Wthree rings^Wcommands to rule^Winstall them all
  - portability, customization
- autotools / autoconf
- CMake, SCons, waf, . . .

```
add_library(myLib a.c b.c)
add_executable(myProg c.c)
target_link_libraries(myProg myLib m)
```

Build systems

- Script
- Makefile
- The nine^Wthree rings^Wcommands to rule^Winstall them all
  - portability, customization
- autotools / autoconf
- CMake, SCons, waf, ...

Windows

- `.sln, .vcproj, MSBuild (Visual Studio)`
- `Cygwin / MinGW`

Tools

- Static code analysis
  - compiler! (`-Wall`)
  - `http://clang-analyzer.llvm.org/`
  - `http://cppcheck.sourceforge.net/`
- Dynamic code analysis
  - simulation / code instrumentation
  - `http://valgrind.org/`
  - `https://github.com/KDE/heaptrack`

Kernel modules

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("James Bond");
MODULE_DESCRIPTION("Hello world.");
MODULE_VERSION("0.0.7");

static char *who = "world";
module_param(who, charp, S_IRUGO);
MODULE_PARM_DESC(who, "Whom to greet");

static int __init hello_init(void){
   printk(KERN_INFO "Hello: Hello %s!\n", who);
   return 0;
}
static void __exit hello_exit(void){
   printk(KERN_INFO "Hello: Goodbye %s!\n", who);
}

module_init(hello_init);
module_exit(hello_exit);
```

Building modules

Makefile

```
obj-m+=hello.o

all:
	make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) clean
```

build & test

```
make
# make install
modinfo hello.ko
insmod hello.ko
insmod hello.ko who=YoYo
# 'modprobe hello' if installed properly
rmmod hello
```