

Unix: kompilator C i narzędzia pomocnicze

Witold Paluszyński

witoldp@pwr.wroc.pl

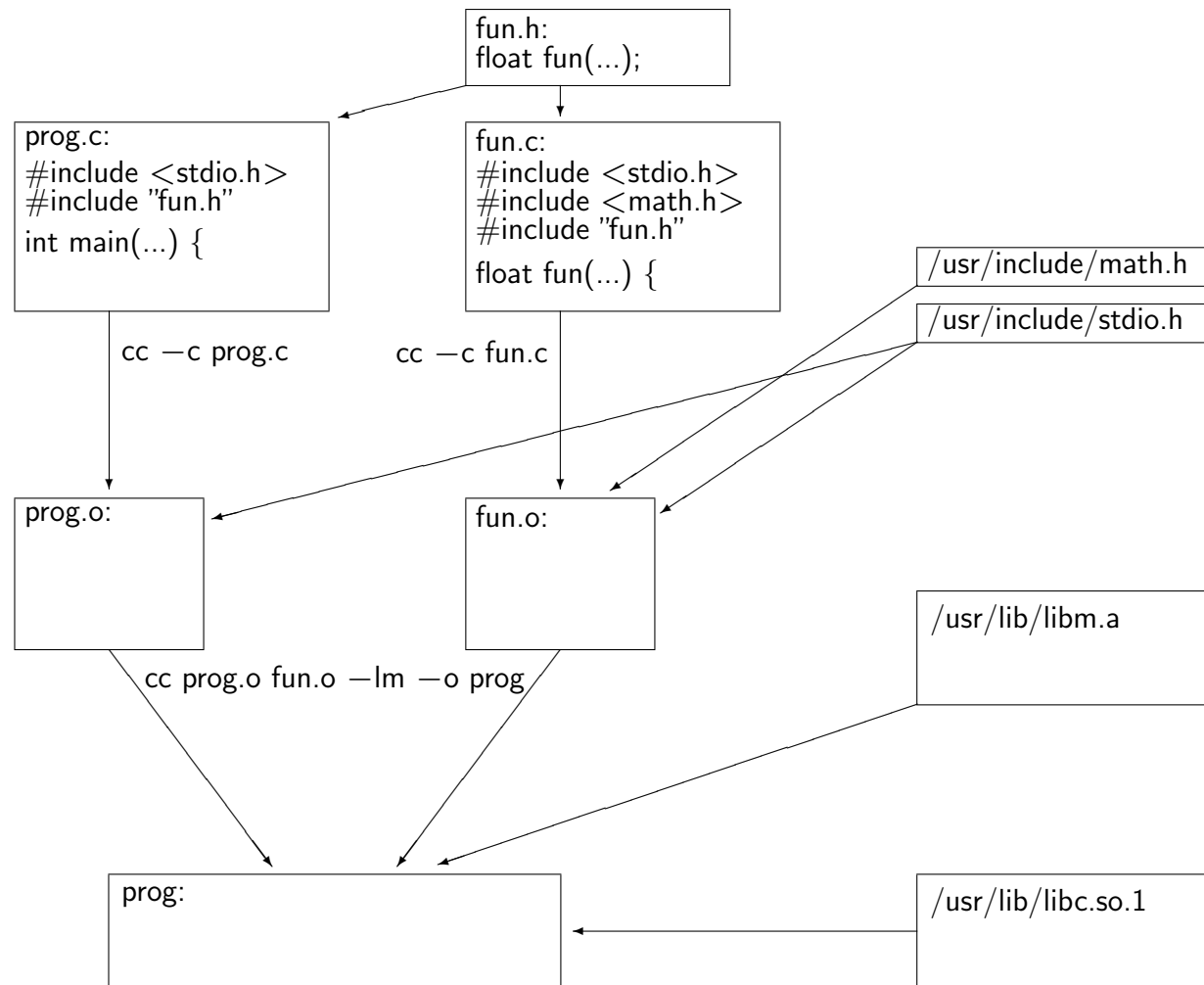
<http://sequoia.ict.pwr.wroc.pl/~witold/>

Copyright © 2001–2006 Witold Paluszyński

All rights reserved.

Niniejszy dokument zawiera materiały do wykładu na temat systemu kompilatora C i narzędzi pomocniczych w systemie Unix. Jest on udostępniony pod warunkiem wykorzystania wyłącznie do własnych, prywatnych potrzeb i może być kopiowany wyłącznie w całości, razem z niniejszą stroną tytułową.

System kompilatora C



wywołanie pełnej kompilacji:

```
cc prog.c fun.c -lm -o prog
```

```
cc -c prog.c
```

```
cc -c fun.c
```

```
cc prog.o fun.o -lm -o prog
```


Opcje wywołania kompilatora C (wspólne)

Tradycyjnie, kompilatory C rozpoznają te opcje jednakowo:

- onazwa** umieść postać wynikową kompilacji w pliku *nazwa*, domyślnie *a.out* dla postaci programu wykonywalnego, *nazwazrodla.s* dla postaci assemblerowej, i *nazwazrodla.o* dla postaci binarnej
- c** pomiń ostatnią fazę kompilacji (linker), nie twórz programu wynikowego, pozostaw postać binarną *.o*
- g** wpisz w program binarny dodatkowe informacje dla debuggera
- lbib** powoduje przeglądanie przez linker biblioteki *bib*, w pliku o nazwie *libbib.a* lub *libbib.so* w kartotece */usr/lib* lub w innych zdefiniowanych ścieżką linkera
- S** wykonaj tylko pierwszą fazę kompilacji do kodu assemblera *.s*
- On** wykonaj optymalizację kodu poziomą *n* (domyślnie poziom 2, który jest na ogół bezpieczny)
- w** pomiń ostrzeżenia (opcja zwykle szkodliwa)

Opcje wywołania kompilatorów (różne)

Niestety, niektóre ważne i pożyteczne opcje występują tylko dla niektórych kompilatorów, lub mają inną postać:

- V** wyświetlaj wywołania kolejnych faz kompilacji (Sun cc)
- v** wyświetlaj wywołania kolejnych faz kompilacji (HP cc, GNU gcc)
- Xc** ściśle przestrzeganie standardu ANSI C (Sun cc)
- Aa** ściśle przestrzeganie standardu ANSI C (HP cc)
- ansi** przestrzeganie standardu ANSI C (GNU gcc)
- pedantic** ściśle przestrzeganie standardu ANSI C (GNU gcc)
- Wall** wyświetlanie ostrzeżeń o wszelkich „dziwnych” konstrukcjach programowych (GNU gcc)

make

skrypt do kompilacji programu:

```
cc -c prog.c
cc -c fun.c
cc prog.o fun.o -lm -o prog
```

specyfikacja Makefile:

```
prog: prog.o fun.o
    cc prog.o fun.o -lm -o prog
prog.o: prog.c fun.h
    cc -c prog.c
fun.o: fun.c fun.h
    cc -c fun.c
```

alternatywna specyfikacja Makefile:

```
CC = gcc
LDFLAGS = -lcurses

prog: prog.o fun.o
prog.o: prog.c
fun.o: fun.c
```

make

make jest uogólnieniem skryptu do kompilacji pakietów wybierającym tylko niezbędne do rekompilacji moduły i właściwe elementy kompilatora:

- ułatwia kompilację pakietów składających się z wielu części, gdzie wprowadzenie poprawek powoduje konieczność rekompilacji niektórych tylko części całego pakietu
- plik opisowy Makefile zawiera specyfikację zależności pomiędzy elementami pakietu, które następnie są sprawdzane przez program make względem dat utworzenia odpowiednich plików
- gdy jakaś zależność nie jest zachowana, make „produkuje” ponownie dany obiekt (plik) według reguł również zawartych w pliku Makefile
- make jest przygotowany do kompilacji programów w C i posiada wbudowany zestaw reguł dla kompilacji tych programów włącznie z całym zestawem narzędzi Unixowych (np. yacc, lex), lecz równie dobrze nadaje się do dowolnego przetwarzania pakietów o dowolnym charakterze (np. skład tekstu skomplikowanego dokumentu) dzięki możliwości zdefiniowania odpowiednich reguł i poleceń

Przykładowy plik Makefile

```
SHELL=/bin/sh
.SUFFIXES: .txt .tex .dvi .ps .ps2 .pdf .bib .bbl .idx .ind

.txt.tex:
    /usr/bin/sed -f $(HOME)/lib/txt2cudzy.sed $< >$@
.tex.dvi:
    latex $<
.dvi.ps:
    dvips -T 11in,8.5in -t landscape $<
.ps.ps2:
    psnup -2 $< $@
    mgland2print $< $@
.dvi.pdf:
    dvi2pdf -l -p letter $*
.bib.bbl:
    bibtex $*
.idx.ind:
    makeindex $<

all: upr_shell upr_tools upr_libs upr_jsys upr_llio upr_proc upr_advio

upr.ps: upr.dvi
upr.dvi: upr.tex
```

pomijamy zaleznosc upr.dvi od upr.bbl i upr.ind bo powoduje petlenie

upr.tex: upr.txt

bib biblio: upr.bbl

upr.bbl: upr.aux Unixprog.bib

 bibtex upr

ind indeks index: upr.ind

upr.ind: upr.idx

upr.idx: upr.tex

upr_shell: upr_shell.ps upr_shell.ps2 upr_shell.pdf

upr_shell.ps: upr_shell.dvi

upr_shell.ps2: upr_shell.ps

upr_shell.pdf: upr_shell.dvi

upr_shell.dvi: upr_shell.tex

upr_shell.tex: upr_shell.txt

upr_tools: upr_tools.ps upr_tools.ps2 upr_tools.pdf

upr_tools.ps: upr_tools.dvi

upr_tools.ps2: upr_tools.ps

upr_tools.pdf: upr_tools.dvi

upr_tools.dvi: upr_tools.tex

upr_tools.tex: upr_tools.txt

```
upr_libs: upr_libs.ps upr_libs.ps2 upr_libs.pdf
upr_libs.ps: upr_libs.dvi
upr_libs.ps2: upr_libs.ps
upr_libs.pdf: upr_libs.dvi
upr_libs.dvi: upr_libs.tex upr_libs_intro.tex upr_libs_string.tex \
               upr_libs_curses.tex upr_libs_search.tex upr_libs_ndbm.tex \
               upr_libs_regcomp.tex upr_libs_crypt.tex
upr_libs.tex: upr_libs.txt
upr_libs_intro.tex: upr_libs_intro.txt
upr_libs_string.tex: upr_libs_string.txt
upr_libs_curses.tex: upr_libs_curses.txt
upr_libs_search.tex: upr_libs_search.txt
upr_libs_ndbm.tex: upr_libs_ndbm.txt
upr_libs_regcomp.tex: upr_libs_regcomp.txt
upr_libs_crypt.tex: upr_libs_crypt.txt

upr_jsys: upr_jsys.ps upr_jsys.ps2 upr_jsys.pdf
upr_jsys.ps: upr_jsys.dvi
               dvips -t a4 upr_jsys
upr_jsys.pdf: upr_jsys.dvi
upr_jsys.dvi: upr_jsys.tex
upr_jsys.tex: upr_jsys.txt

upr_llio: upr_llio.ps upr_llio.ps2 upr_llio.pdf
upr_llio.ps: upr_llio.dvi
```

```
        dvips -t a4 upr_llio
upr_llio.pdf: upr_llio.dvi
upr_llio.dvi: upr_llio.tex
upr_llio.tex: upr_llio.txt

upr_advio: upr_advio.ps upr_advio.ps2 upr_advio.pdf
upr_advio.ps: upr_advio.dvi
        dvips -t a4 upr_advio
upr_advio.pdf: upr_advio.dvi
upr_advio.dvi: upr_advio.tex
upr_advio.tex: upr_advio.txt

upr_proc: upr_proc.ps upr_proc.ps2 upr_proc.pdf
upr_proc.ps: upr_proc.dvi
        dvips -t a4 upr_proc
upr_proc.ps2: upr_proc.ps
        psnup -pa4 -Pa4 -2 upr_proc.ps upr_proc.ps2
upr_proc.pdf: upr_proc.dvi
upr_proc.dvi: upr_proc.tex
upr_proc.tex: upr_proc.txt

upr_thr: upr_thr.ps upr_thr.ps2 upr_thr.pdf
upr_thr.ps: upr_thr.dvi
        dvips -t a4 upr_thr
upr_thr.ps2: upr_thr.ps
```

```
psnup -pa4 -Pa4 -2 upr_thr.ps upr_thr.ps2  
upr_thr.pdf: upr_thr.dvi  
upr_thr.dvi: upr_thr.tex  
upr_thr.tex: upr_thr.txt
```


System kontroli wersji RCS

Funkcje systemu RCS (Revision Control System):

- pamiętanie pełnego zestawu archiwalnych wersji danego pliku z automatyczną numeracją wersji, dokumentacją i rejestracją czasu utworzenia i autora; dostęp do dowolnej wersji poprzez: numer, datę, autora, bądź nazwę,
- obsługa dodatkowych (bocznych) odgałęzień wersji pliku,
- koordynacja pracy pomiędzy wieloma programistami pracującymi nad projektem poprzez ustawianie praw dostępu oraz mechanizm wypożyczania i zwracania plików,
- ograniczona możliwość automatycznego łączenia dwóch różnych zmodyfikowanych wersji jednego pliku; powstanie takich wersji może być wynikiem nieporozumienia programistów, bądź chęci przeniesienia na jedną gałąź rozwojową pliku poprawek dokonanych na innej gałęzi
- korzystanie z automatycznie aktualizowanych znaczników w plikach.

System RCS: polecenia

- `ci` przekazuje plik systemowi RCS, nadaje mu numer wersji (kolejny) oraz pobiera i prosi o podanie opisu dokonanej modyfikacji; oryginalny plik zostaje skasowany (z wyjątkiem `ci -u` lub `ci -l`, to ostatnie wywołanie od razu blokuje plik do edycji)
- `co` pobiera plik z systemu RCS; z opcją `-l` również blokuje plik tak, że wypożyczoną kopię można edytować
- `rlog` wyświetla kompletną historię pliku włącznie z opisami istniejących wersji i aktualnym stanem wypożyczenia pliku
- `rcsdiff` wyświetla różnice pomiędzy bieżącym plikiem a jego ostatnią wersją przechowywaną przez system RCS
- `rmerge` służy do połączenia dwóch zmodyfikowanych wersji jednego pliku
- `rcs` – komenda administracyjna, służy do modyfikacji ustawień

System RCS: przykład

```
shasta-201> ci upr_shell.txt
upr_shell.txt,v <-- upr_shell.txt
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> Prezentacja na temat Bourne shella i filtrow tekstowych
>> .
initial revision: 1.1
done
shasta-202> ls -l upr_shell.txt
upr_shell.txt: No such file or directory
shasta-203> co -l upr_shell.txt
upr_shell.txt,v --> upr_shell.txt
revision 1.1 (locked)
done
shasta-206> ls -l upr_shell*
-rw-r--r--  1 witold  gurus      180 Nov  5 11:58 upr_shell.aux
-rw-r--r--  1 witold  gurus    41556 Nov  5 10:59 upr_shell.dvi
-rw-r--r--  1 witold  gurus     9681 Nov  5 11:58 upr_shell.log
-rw-r--r--  2 witold  gurus   129537 Nov  5 11:58 upr_shell.pdf
-rw-r--r--  2 witold  gurus   215917 Nov  5 10:59 upr_shell.ps
-rw-r--r--  1 witold  gurus   243737 Nov  5 10:59 upr_shell.ps2
-rw-r--r--  1 witold  gurus    30300 Nov  5 10:59 upr_shell.tex
-rw-r--r--  1 witold  gurus    30300 Nov 12 15:17 upr_shell.txt
-r--r--r--  1 witold  gurus    30555 Nov 12 15:17 upr_shell.txt,v
```

```
shasta-207> rlog upr_shell.txt
```

```
RCS file: upr_shell.txt,v
```

```
Working file: upr_shell.txt
```

```
head: 1.1
```

```
branch:
```

```
locks: strict
```

```
access list:
```

```
symbolic names:
```

```
keyword substitution: kv
```

```
total revisions: 1;      selected revisions: 1
```

```
description:
```

```
Prezentacja na temat Bourne shella i filtrow tekstowych
```

```
-----
```

```
revision 1.1      locked by: witold;
```

```
date: 2003/11/12 14:17:50;  author: witold;  state: Exp;
```

```
Initial revision
```

```
=====
```

Często użycie systemu RCS polega na utworzeniu podkartoteki o nazwie RCS, w której polecenia systemu RCS umieszczają swoje archiwa, co pozwala np. na łatwą archiwizację. Często również każde wprowadzenie pliku do archiwum od razu związane jest z zatrzymaniem jego oryginalnej wersji źródłowej do dalszej edycji przez autora: `ci -l upr_shell.txt`

Linker: tworzenie bibliotek dynamicznych

```
/* file shrojb.c */  
const char *myfunc() {return "Hello World";}
```

```
/* file hello.c */  
#include <stdio.h>  
extern const char *myfunc();  
main() {  
    printf("%s\n", myfunc());  
    return 0;  
}
```

```
$ gcc -fpic -c shrojb.c  
$ gcc -shared -o libshared.so shrojb.o  
$ gcc hello.c libshared.so  
$ ./a.out  
Hello World
```

- AIX 3.2 using xlc (unverified):
Drop the '-fpic', and use '-bM:SRE -bE:libshared.exp' instead of '-shared'.
You also need to create a file 'libshared.exp' containing the list of symbols to export, in this case 'myfunc'. In addition, use '-e _nostart' when linking the library (on newer versions of AIX, I believe this changes to '-bnoentry').

- SCO OpenServer 5 using the SCO Development System (unverified):
Shared libraries are only available on OS5 if you compile to ELF format, which requires the '-belf' option. Use '-Kpic' instead of '-fpic', and 'cc -belf -G' for the link step.
- Solaris using SparcWorks compilers:
Use '-pic' instead of '-fpic', and use 'ld -G' instead of 'gcc -shared'.

Other issues to watch out for:

- AIX and (I believe) Digital Unix don't require the -fpic option, because all code is position independent.
- AIX normally requires that you create an 'export file', which is a list of symbols to be exported from the shared library. Some versions of the linker (possibly only the SLHS linker, svld?) have an option to export all symbols.
- If you want to refer to your shared library using the conventional '-l' parameter to the linker, you will have to understand how shared libraries are searched for at runtime on your system. The most common method is by using the LD_LIBRARY_PATH environment variable, but there is usually an additional option to specify this at link time.

- Most implementations record the expected runtime location of the shared library internally. Thus, moving a library from one directory to another may prevent it from working. Many systems have an option to the linker to specify the expected runtime location (the '-R' linker option on Solaris, for example, or the LD_RUN_PATH environment variable).
- ELF and a.out implementations may have a linker option '-Bsymbolic' which causes internal references within the library to be resolved. Otherwise, on these systems, all symbol resolution is deferred to the final link, and individual routines in the main program can override ones in the library.