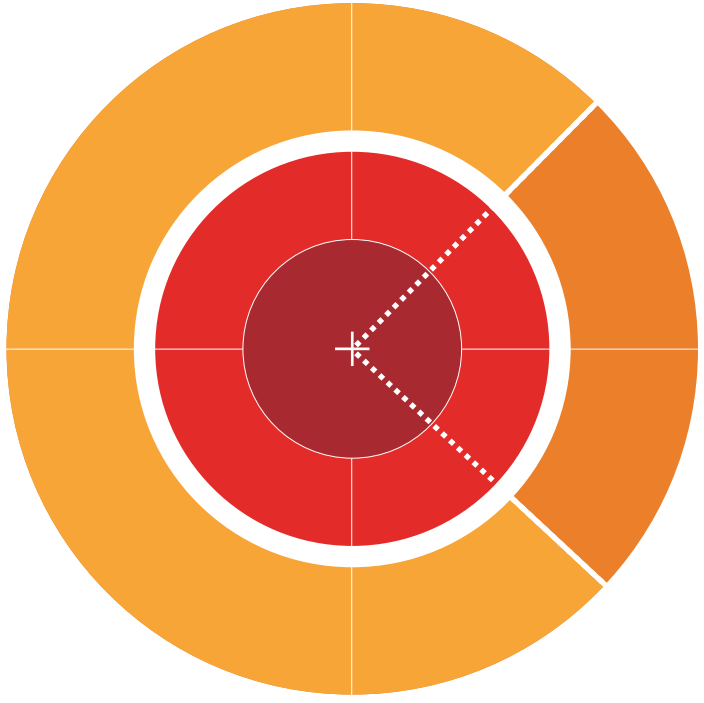
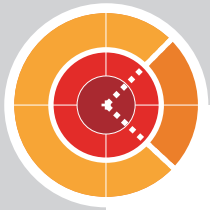


Řízený překlad - make

- nástroj **`make`** je původně UNIXový pomocný vývoj. nástroj vzniklý v AT&T asi 1975
- **účel**: zjednodušit a zautomatizovat překlad a sestavování (linking) větších projektů, kde není zjevné např. v jakém pořadí je třeba ZK překládat
- zároveň částečně dokumentuje mechanismus sestavení projektu (závislosti modulů, atp.) a umožňuje provést např. migraci a instalaci
- lze programovat bez **`make`**, ale **není to dobrý nápad**
- překlad pomocí **`make`** je řízen souborem **`makefile`**
- **`makefile`** je prostý textový soubor (ASCII), který obsahuje (i) **komentáře**, (ii) **pravidla** *(rules)*, (iii) **cíle** *(targets)*, (iv) **makra** a (v) **příkazy** *(commands)* zapsané v předem dané podobě



Komentáře v Makefile

- komentář začíná znakem **#** kdekoliv na řádce a končí koncem řádky

```
# The BIN and LIB macros define the
# output directories for binaries and
# compiled units.

DCC = dcc32 -q # set the C compiler
TASM = tasm32
BRCC = brcc32
...
```

- zvláště složitější makefile je **nutné dobře komentovat**, aby plnil i svojí "dokumentační" funkci

Pravidla (rules)

- pravidla **definují postup**, který vede ke splnění daného cíle (např. přeložení ZK do `.obj` nebo sestavení `.exe`)
- obecný tvar pravidla je

<cíl>: <řádka závislostí>
TAB *<příkazová řádka>*

```
file1.exe: file1.obj
TAB link386 file1.obj
...
```

tento soubor je třeba ke splnění cíle **`file1.exe`**

```
target1: dep1 dep2 dep3 dep4
target1: dep5 dep6 dep7
        wcc386 $**
```

k jednomu cíli může existovat více řádek závislostí, ale jen jediná musí obsahovat příkaz. řádku

`$**` se nahradí jmény souborů v řádce závislostí

Explicitní pravidla

- pravidla, která **make** používá při plnění cílů

```
file1.exe: file1.obj \
    file2.obj
    link386 file1.obj file2.obj
...

mylib.lib:: f1.obj f2.obj
    echo Adding C files
    wlib mylib +f1.obj +f2.obj

mylib.lib:: f3.obj f4.obj
    echo Adding ASM files
    wlib mylib +f3.obj +f4.obj
```

pokračuje na další řádce

daný cíl je definován více než jedním pravidlem

Implicitní pravidla

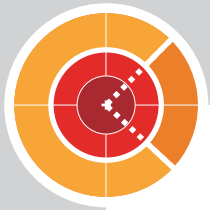
- obecná pravidla, která definují, jak vznikají soubory s určitými příponami

přípona zdrojového souboru
přípona cílového souboru

} může předcházet cesta k adresáři, kde se hledají zdrojové soubory, resp. ukládají cílové

toto makro nahrazuje **myprog.obj** za **myprog.c**

místo příkazů jen prázdná řádka, protože **make** už umí díky předchozímu implicitnímu pravidlu překládat **.c** do **.obj**



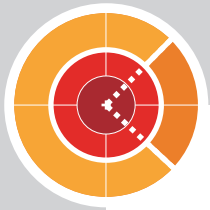
Cíle (targets) - výsledné akce a zkompilované soubory

- cíl určuje, co je výsledkem činnosti **`make`** (např. **`.exe`**)
- v **`makefile`** může být definováno množství cílů, nesmí být stejně pojmenovány

```
file1.exe: file1.obj
    bcc32 file1.obj
file2.exe: file2.obj
    bcc32 file2.obj
```

definice cíle je tvořena pravidlem (viz předchozí výklad) - v této ukázce je **cílem** vytvoření souboru **`file1.exe`**

- je-li v řádce závislostí soubor (resp. cíl), který neexistuje, hledá se pravidlo, které ho vytvoří (pokud neexistuje, hlásí **`make`** chybu)
- cílem **nemusí být** nutně jen **soubor** (viz dále tzv. symbolické cíle)



Symbolické cíle (symbolic targets)

- symbolický cíl umožňuje vykonat několik akcí současně, např. vytvořit několik spustitelných souborů
- v **`makefile`** může být definováno více symbolických cílů, **`make`** vytvoří buď ten, který uvedeme na příkazové řádce, nebo **první**

```
AllFiles: file1.exe file2.exe
file1.exe: file1.obj
    bcc32 file1.obj
file2.exe: file2.obj
    bcc32 file2.obj
```

symbolický cíl - nebude vytvořen soubor **`AllFiles`**, ale soubory **`file1.exe`**, **`file2.exe`**

- po řádce definující **symbolický cíl** nenásledují příkazy
- **symbolický cíl** musí mít jedinečné jméno (soubor takového jména nesmí v adresáři existovat, jméno odpovídá požadavkům OS na pojmenovávání souborů)

Explicitní určení symbolického cíle

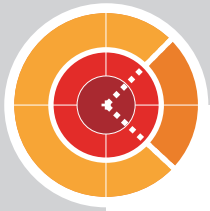
- parametrem **`make`** lze určit, který symbolický cíl se má provést, např. **`C:\>make all`** nebo **`C:\>make clean`**

```
…
all: train test gen_result
…
clean:
      echo A | del *.*
…
```

žádné závislosti, smazat soubory lze vždy (nic k tomu není potřeba)

příkazy OS, které se mají provést, aby byl cíl splněn

- uživatel **nemusí parametr** na příkazové řádce **uvést** => na prvním místě musí být správný symbolický cíl (zde cíl **`all`**)



Pořadí cílů - na pořadí záleží

```
test.obj: test.c
      wcc386 test.c

test.exe: test.obj
      link386 test.obj
```

```
test.exe: test.obj
      link386 test.obj

test.obj: test.c
      wcc386 test.c
```

Otázka: Který z těchto dvou **`makefile`** vytvoří spustitelný soubor **`test.exe`**, vzniklý překladem ZK **`test.c`**?

Pořadí cílů - na pořadí záleží

```
test.obj: test.c
      wcc386 test.c

test.exe: test.obj
      link386 test.obj
```

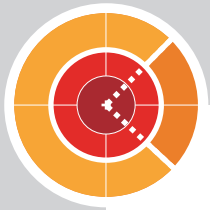
soubor **`test.c`** je k dispozici, není tedy třeba hledat další cíle...

```
test.exe: test.obj
      link386 test.obj

test.obj: test.c
      wcc386 test.c
```

soubor **`test.obj`** neexistuje - **`make`** hledá pravidlo na jeho vytvoření...

Odpověď: Samozřejmě, že ten **druhý** - neuvede-li uživatel jinak, plní se první nalezený cíl, což je v prvním případě pouze překlad do **`.obj`**



Makra v makefile

<jméno>=<text makra>

- **make** rozlišuje velká/malá písmena => **MyMacro** a **mymacro** jsou rozdílná makra

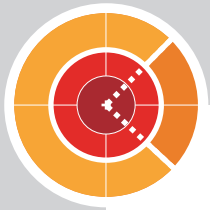
```
MYEXT=.c
SOURCE=f1.c f2.c f3.c
```

okolo = nejsou dovolené mezery

- jméno makra max. 512 znaků, text makra max. 4096 znaků
- makro lze nadefinovat i při volání **make** z příkazové řádky

```
make -Dcommand="wcc386 -fp5"
```

- je-li makro stejného jména definováno v **makefile**, použije se to z **makefile**

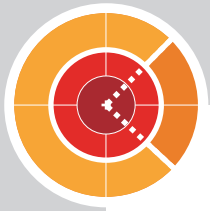


Vestavěná makra

nespoléhat na tato makra - liší se podle verze `make`

`AS`	assembler
`CC`	překladač C
`MAKE`	příkaz, kterým byl spuštěn **`make`**
`MAKEFLAGS`	parametry **`make`** z příkazové řádky

Makro	v implicitním	v explicitním
`$*`	cesta\závislý	cesta\cílový
`$**`	cesta\závislý+příp	vše-závislé+příp
`$<`	cesta\závislý+příp	cesta\cílový+příp
`$?`	cesta\závislý+příp	staré-závislé
`$&`	závislý	cílový
`$.`	závislý+příp	cílový+příp
`$:`	cesta-k-závislému	cesta-k-cílovému
`$@`	cesta\cílový+příp	cesta\cílový+příp

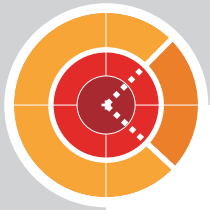


Ukázka použití vestavěných maker

```
CC=wcc386
CFLAGS=-fp5 -mf -ob,l
...
.c.obj:
    $(CC) -c $(CFLAGS) $(INCLUDES) $**
```

Přesnou podobu vestavěných maker je nutné dohledat v dokumentaci použité verze `make`. Makra se mohou dramaticky lišit, např. mezi GCC a Microsoft Visual C/C++.

	GCC	Watcom
Úplné jméno souboru ze závislostí	`$<`	`$**`



Příkazy v makefile

- příkazy OS

```
clean:
    @echo Cleaning...
    @echo Y | del *.*
```

příkaz se provede, ale nevypíše do konzole

```
install: main.exe
    @echo Installing...
    @ren main.exe win.exe
    @copy win.exe C:\Windows
```