

Preklad isu-cv9-1.2sm:

1) SASM / Windows :

termkeys.inc skopírovať do SASM/Windows/includes/

2) Eclipse / Windows :

termkeys.inc skopírovať do zložky projektu (k isu-cv9-1.2sm)

3) CMD / Windows :

termkeys.inc skopírovať do zložky k isu-cv9-1.2sm

4) Linux :

získať knižnicu curses :

a) štolské počítače

termkeys.inc a knižnice z archívu centos68-curses32.tar.gz
pridať do zložky k zdrojovému kódu

b) Všet Linux : nainštalovať 32-bit vývojársku verziu libcurses
alebo vyškúšať centos68-curses32.tar.gz

prilinkovať k programu (LDFLAGS v Makefile, eclipse alebo ručne)

a) statické knižnice : *.a -ldl

b) z balíčka distribúcie : -lcurses

tzn. linker by mal byť spustený asi nasledovne :

```
gcc -m32 isu-cv9-1.0 *.a -ldl
```

resp.

```
gcc -m32 isu-cv9-1.0 -lcurses
```

Spustenie isu-cv9-1

1) SASM / Windows : "Debug" → "Run in new window"

2) Eclipse : spustiť binárku priamo z umiestnenia projektu

Ladenie isu-cv9-1 (veľa šťastia)

1) Windows : x32-dbg, v kóde je aktívne íchzenie - je dobré si rozumne vyrobiť breakpoint na main

2) Linux : gdb

A) Switch-case

① postupným porovnáním

if - else if - else if - ... - else

```
.case1:
  cmp [ebx], 1
  jne .case2
  :
  jmp .end
.case2:
  cmp [ebx], 2
  jne .case3
  :
  jmp .end
:
.default:
  :
.end:
```

② tabulkou hodnot a adres

```
section .data
  table dd 1, case1
         dd 2, case2
         :
         :
         :
```

Section .text

```
mov ecx, [pocet_moznosti]
cyclics:
  # porovnat hodnotu
  # skocit
  loop cyclics
  jmp default
case1:
  :
  jmp .end
case2:
  :
  jmp end
default:
  :
  end
```

③ "lookup table"

- každý vieme mapovať vstup na indexy
≈ 10 položiek, "husto" za sebou

Section .data

```
table dd case1
      dd default
      dd case3
      dd case4
```

Section .text

```
sub eax, 1
jmp [table + 4 * eax]
```

```
case1:
  :
  jmp end
  :
```

Ⓑ vyhodnocovanie podmienok (zložených)

- skratové: `if (ptr && *ptr == 1)`

```
test eax, eax
jz .end
mov ecx, [eax]
cmp ecx, 1
jne .end
:
.end:
```

- jeden slok - nie vždy možné (zlé prístupy)

`if (eax && ebx)`

```
test eax, eax
setnz al
test ebx, ebx
setnz ah
test al, ah
jz .end
:
.end:
```

Setcc použiteľné tiež
na vyhodnotenie výrazu

```
int x = a == b;
```

štandard C garantuje, že
x bude 0 alebo 1