PB161

8. cvičení

Lukáš Ručka

11. listopadu 2011

Obecné pojmy

- *breakpoint*
  místo v programu, kde se má jeho běh zastavit
- *watch*
  místo v paměti, při jehož manipulaci se má běh zastavit

Překlad a spuštění

Nutno upravit kompilační příkaz:

```
g++ –g –ggdb3 –O0 –c zdrojX.cpp –o zdrojX.o
```

Překlad a spuštění

Nutno upravit kompilační příkaz:

```
g++ –g –ggdb3 –O0 –c zdrojX.cpp –o zdrojX.o
```

- **–g** povolí generování ladících symbolů
- **–ggdb3** generuje ladící symboly ve formátu vhodném pro gdb (volitelný)
- **–O0** vypne optimalizace (optimalizace činí některé hodnoty nedosažitelnými)

Překlad a spuštění

Nutno upravit kompilační příkaz:

```
g++ –g –ggdb3 –O0 –c zdrojX.cpp –o zdrojX.o
```

- **–g** povolí generování ladících symbolů
- **–ggdb3** generuje ladící symboly ve formátu vhodném pro gdb (volitelný)
- **–O0** vypne optimalizace (optimalizace činí některé hodnoty nedosažitelnými)

Pozn.: Na aise chybí knihovny s ladícími informacemi. Pro ladění tedy můžete použít některou z nymf, kde se tyto vyskytují.

Překlad a spuštění

Následně debugger spustíme (ladíme program *./foo* s argumenty *–baz*):

```
gdb --args ./foo --baz
```

Překlad a spuštění

Následně debugger spustíme (ladíme program *./foo* s argumenty *–baz*):

```
gdb --args ./foo --baz
```

Korektní stav – (gdb) označuje prompt debuggeru):

```
Reading symbols from ./foo...done.
(gdb) _
```

Překlad a spuštění

Přidáme úvodní breakpoint a program spustíme

```
(gdb) break main
Breakpoint 1 at 0x400ca4: file main.cpp,
    line 16.
(gdb) run
Starting program: /home/xrucka/pb161/cv8/
    foo
Breakpoint 1, main () at main.cpp:16
16          int x = 0;
(gdb) _
```

Překlad a spuštění

Přidáme úvodní breakpoint a program spustíme

```
(gdb) break main
Breakpoint 1 at 0x400ca4: file main.cpp,
    line 16.
(gdb) run
Starting program: /home/xrucka/pb161/cv8/
    foo
Breakpoint 1, main () at main.cpp:16
16          int x = 0;
(gdb) _
```

Pakliže gdb vypíše informace o chybějících ladících symbolech, je nutno toto doinstalovat (pakliže máte oprávnění), nebo zkusit jiný stroj.

```
Missing separate debuginfos...
```

Ladění - základ

- **run** - spustí laděný program
- **quit** - ukončí laděný program
- **break** kde - nastaví breakpoint na *kde*
- **watch** co - nastaví watchpoint na *co*
- **file** binarka - (znovu) načte program jménem *binarka*

Ladění - krokování

- **continue** - pokračuje k dalšímu breakpointu
- **next** - posune se na další řádek v současném souboru
- **step** - vstoupí do volaného řádku
- **finish** - dokončí proud instrukcí v rámci zásobníku (tj. až po výstup z funkčního volání - známé také jako *step-out*)

Ladění - navigace

- **list** - vypíše aktuální oblast v kódu
- **print** co - vypíše proměnnou či výraz *co*
- **info stack** - vypíše stav zásobníku (volání)
- **info break** - vypíše breakpointy či watches

Ladění - breakpointy 2

- **enable** id - povolí breakpoint *id*
- **disable** id - zakáže breakpoint *id*
- **del** id - smaže breakpoint či watchpoint *id*

Příklady - print

```
(gdb) print x
$1 = 0
(gdb) print x+3
$2 = 3
(gdb) print ptr
$3 = (int *) 0x7fffffffd9f0
(gdb) print *ptr
$4 = 3
```

Příklady - break

```
(gdb) break main
Breakpoint 1 at 0x4005e3: file main.cpp,
    line 16.
(gdb) break main.cpp:5
Breakpoint 2 at 0x4005ea: file main.cpp,
    line 5.
(gdb) break main.cpp:7 if y == 3
Breakpoint 3 at 0x4005f1: file main.cpp,
    line 7.
```

Rady na závěr

- Ladící výpisy jsou mnohdy užitečnější než krokování
- Pokud program běží, gdb není vidět - v takovém případě pomůže CTRL+C
- Laděni způsobuje těžkou synchronizaci, stejně tak výpisy - vlákna se tedy ladí velmi obtížně