




Razhroščevalnik



Seminarska naloga pri predmetu SPO
Jure Jesenšek, januar 2015

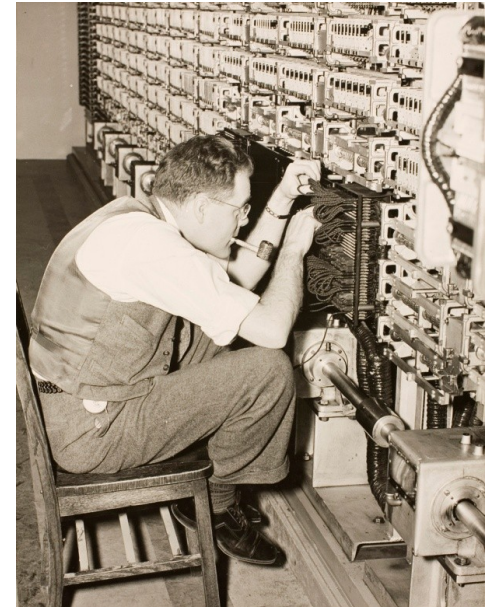
Uvod

- ▶ Delovanje razhroščevalnikov
- ▶ Strojna in programska podpora razhroščevanju
- ▶ Proces razhroščevanja
- ▶ Nekaj najbolj popularnih razhroščevalnikov
- ▶ Primer uporabe GNU Debugger



Splošno

- ▶ Razhroščevalnik – program za testiranje drugih programov
- ▶ Zgodovina izrazov hrošč (bug) in razhroščevanje (debugging)
- ▶ Osnoven postopek razhroščevanja
- ▶ Pogoste tehnike:
 - ▶ Print debugging
 - ▶ Post-mortem
 - ▶ “Wolf fence”
 - ▶ Delta debugging
 - ▶ ...



Način delovanja

- ▶ Past (trap) - deljenje z nič, nedovoljen pomnilniški dostop ali pa breakpoint
- ▶ Predaja izvajanja na CPE k OS ali razhroščevalniku
- ▶ Pregled kode:
 - ▶ Source-level oz. symbolic debugger – (v IDE-jih) pokaže vrstico ustavitve v prvotni izvorni kodi
 - ▶ Low-level oz. machine language debugger - pokaže vrstico ustavitve v disassemblyju



Strojna podpora

- ▶ Tehnologije, vgrajene v samo CPE
 - ▶ Single-stepping (trap flag)
 - ▶ Podpora za HW breakpoint-e (primerjalniki)
 - ▶ Nabor ukazov za virtualizacijo
 - ▶ In-system programming – reprogramiranje namesto menjavanje čipov



Prekinitvene točke (breakpoint)

- ▶ Lokacija, kjer želimo da se program ustavi
- ▶ Instruction breakpoint
- ▶ Conditional/data breakpoint (watchpoint)
- ▶ Hardware in software breakpoint



Strojne prekinitvene točke

- ▶ Namenski registri na CPE (DR – debug register)
- ▶ Odvisni od arhitekture CPE
- ▶ x86:
 - ▶ 6 registrov za 4 strojne prekinitvene točke
 - ▶ DR0-DR3 = linearni naslovi prekinitvene točke
 - ▶ DR6 in DR7 = statusi prekinitev in nadzor strojnih prek. Točk
- ▶ Pogosto kot “pomnilniške” prekinitvene točke
 - prekinitev pri dostopanju/pisanju v pomnilnik
- ▶ Tudi pri razhroščevanju I/O naprav

Programske prekinitvene točke

- ▶ Pogostejše kot strojne
 - ▶ Zamenjava opcode-a s pastjo
 - ▶ Pri izvedbi debugger ustavi ciljni program
 - ▶ Obnovi prvoten opcode
 - ▶ Izvede ukaz
 - ▶ Zopet vstavi past
-
- ▶ Star ukaz v cevovodu – izprazni cevovod in ponovno prevzame ukaz
-
- ▶ Problemi z dostopom, velikostjo pasti, počasnostjo



Razhroščevanje v Javi

- ▶ Precej drugačno
- ▶ Ni naslovov
- ▶ Ni strojne kode

- ▶ Java Virtual Machine Tool Interface
 - ▶ Storitve, ki jih omogoča JVM v namen razhroščevanja
 - ▶ Zahteve za informacije (stanje spremenljivk)
 - ▶ Akcije (postavljanje prek. točk)
 - ▶ Obvestila (program doseže prek. točko)



Seznam razhroščevalnikov

- ▶ Razhroščevalniki
 - ▶ GDB (GNU)
 - ▶ LLDB (LLVM)
 - ▶ MS Visual Studio Debugger
 - ▶ WDW (OpenWatcom)
 - ▶ dbx (Berkeley Unix)
 - ▶ Intel debugger
 - ▶ Valgrind (memory debugging)
 - ▶ WinDbg (Windows)
 - ▶ CodeView (MS-DOS)
 - ▶ sdb (stari UNIX)
 - ▶ jdb (Java)
 - ▶ Python debugger
 - ▶ DBG (PHP)
 - ▶ ARM DS-5
- ▶ GUI front-end:
 - ▶ DDD (GDB, jdb, python, Perl...)
 - ▶ Emacs (GDB)
 - ▶ xxgdb (X-window in dbx)
 - ▶ Qt Creator (GDB, CDB, LLDB)
 - ▶ Eclipse
 - ▶ CodeBlocks
- ▶ ...in seveda mnogi drugi



GDB

- ▶ GNU Debugger - del GNU
- ▶ Richard Stallman
- ▶ Na praktično vseh GNU/Linux OS
- ▶ Na začetku temeljil na DBX
- ▶ Ada, C, C++, Objective-C, Pascal, Fortan, Java, delna podpora ostalim
- ▶ ARM, x86 in x86-64, Itanium, PowerPC, Motorola 68000, itd.
- ▶ Sledenje, spreminjanje izvajanja programa
- ▶ Klic funkcij neodvisno od izvajanja programa
- ▶ Remote debugging via TCP/IP (za vgrajene sisteme, Linux kernel)
- ▶ Brez GUI (DDD, xgdb, kdbg...)
- ▶ IDEji: CodeBlocks, Geany, Qt Creator, Eclipse, NetBeans, MS Visual Studio in Emacs



Prikaz uporabe GDB

- ▶ `gcc -g -o imeprograma program.c`
- ▶ Osnove: `segafult.c`
- ▶ Priklop na tekoč proces: `sleep.c`

