Narzędzia do debugowania

Jakub Salamon

Heaptrack

- zużycie pamięci programu
- znalezienie punktów aktywnych, które należy zoptymalizować
- znalezienie wycieków pamięci
- znalezienie miejsc, w których jest przydzielane najwięcej pamięci
- znalezienie tymczasowych alokacji, które są poprzedzone zwolnieniem pamięci
- szybszy niż valgrind

sudo apt-get install -y heaptrack

Minusy

- obsługuje tylko stertę
- tylko linux
  - winheaptrack

Dstat

- służy do monitorowania wydajności systemu
  - użycie procesora
  - użycie dysków
  - stanu sieci
  - stanu zarządzania pamięcią
- ma możliwość pokazania przerwań urządzeń

Opcje uruchomienia

- -c / --cpu statystyki procesora
- -d / --disk statystyki dysku
- -n / --net statystyki sieci
- -g / --page statystyki pamięci swap
- -y / --sys statystyki systemowe, przerwania, przełączniki kontekstu
- -i / --int statystyki przerwań

Domyślnie

- -a
- --all
- -cdngy

Perf

- narzędzie do analizy wydajności w systemach linux
- może statystycznie profilować cały system
- obsługuje sprzętowe liczniki wydajności, punkty śledzenia, liczniki wydajności oprogramowania i sondy dynamiczne

Podkomendy

- list - wypisuje listę dostępnych zdarzeń
- record - pomiar i zapis danych próbkowania dla danego programu
- report - generowanie profilu płaskiego lub wykresu
- trace - strace
- stat - mierzy całkowitą liczbę zdarzeń pojedynczego programu

ALD - assembly language debugger

- d - deasemblacja
- n - wykonywanie krok po kroku
- break - ustawienie breakpointa
  - break 0x80480D1
- run - uruchomienie programu
- continue - wznowienie działania po breakpoincie

```
080480C2:<_start>    B807000000    mov eax, 0x7
080480C7             E80C000000    call near +0xc (0x8048096:_printEAXdecimal)
080480CC             B801000000    mov eax, 0x1
080480D1             BB05000000    mov ebx, 0x5
080480D3             CD80          int 0x80
```

GDB

- duży rozwój na przestrzeni ostatnich lat
- ma łatwo dostępną dokumentację
- jest prosty w użytkowaniu

Dlaczego asembler w gdb?

- mamy plik binarny i kod, ale nie możemy go ponownie skompilować
- mamy problem z błędem kompilacji wersji i nie mamy dostępu do flag kompilacji
- mamy problem z biblioteką zewnętrzną

Catch 2

- biblioteka do pisania testów jednostkowych w języku C++
- prosta w obsłudze
- prosta w instalacji

Instalacja

- git clone https://github.com/catchorg/Catch2.git
- cd Catch2
- cmake -Bbuild -H. -DBUILD_TESTING=OFF
- sudo cmake --build build/ --target install

```
#include <catch2/catch.hpp>
```

Matchers CHECK_THAT(sth, *)

- Contains(str)
- StartsWith(str)
- EndsWith(str)
- VectorContains(value)

TEST_CASE(“Matchers”, “[matchers][operators][operator||][operator&&]”)

REQUIRE_THAT(func(), Contains(“word”) && ((StartsWith(“begin”) || StartsWith(“start”))

Dziękuję za uwagę