g++, gdb, valgrind

Wykład 3

GCC

- W skład GCC wchodzą kompilatory następujących języków programowania
  - C – gcc
  - C++ - g++
  - Objective-C – gobjc
  - Fortran - g77 oraz nowa implementacja Fortranu 95 o nazwie Gfortran
  - Java – gcj
  - Ada – gnat
- Kompilatory wchodzące w skład GCC mogą być uruchamiane na wielu różnych platformach sprzętowych i systemowych
- Za ich pomocą można generować kod wynikowy przeznaczony dla różnych procesorów i systemów operacyjnych
  - Wraz z tzw. kompilacją skrośną (cross-compilation)

Budowa i działanie

- Kompilator *GCC* składa się z 3 głównych części
  - front end
  - middle end
  - back end
- Front end
  - Oddzielny dla każdego języka
  - Powstaje w jego wyniku „abstract syntax tree”
  - A następnie powstaje **GENERIC** i **GIMPLE**
- Middle end
  - Optymalizacja kodu – usuwanie „martwego” kodu, obliczanie stałych itp.
- Back end
  - Generuje kod asemblera dla konkretnej architektury
  - Dokonywane są dalsze optymalizacje z uwzględnieniem możliwości CPU

Podstawowe opcje

- -c – nie wykonuj konsolidacji
- -S – generuje kod w asemblerze
- -S -masm=intel – generuje kod w asemblerze (składnia intel)
- -E – zatrzymuje się po preprocesingu
- -o FILE – definiuje wyjściową nazwę pliku
- -pipe – używa potoków zamiast plików tymczasowych
- -x – można określić język jaki ma być użyty do kompilacji danego kodu
- --help

Opcje języka c++ (nie tylko)

- -g – produkuje informacje do debugowania
- -pg – generuje informacje dla profilera gprof
- -std – wybiera standard języka [c++98, c++03, c++11, …]
- -fsyntax-only – sprawdza tylko składnie
- -fno-rtti – wyłącza RTTI czyli dynamic_cast i typeid
- -fno-default-inline – funkcje zdefiniowane w ciele klasy nie będą domyślnie inline
- -Wnon-virtual-dtor (C++ only)
- -Wall
  - Włącza praktycznie wszystkie błędy
- -pedantic (-pedantic-errors)
  - Zgłasza wszystkie ostrzeżenia/błędy zgodnie z daną specyfikacją języka
- -O[0123s] – wykonuje optymalizację
- -D name – definuje nazwę
- -D name=val – definiuje nazwę z wartością
- -Idir – pozwala określić ścieżkę do plików nagłówkowych
- -Ldir – j.w. ale biblioteki
- -llib – dołącza bibliotekę do linkera
- https://gcc.gnu.org/onlinedocs/

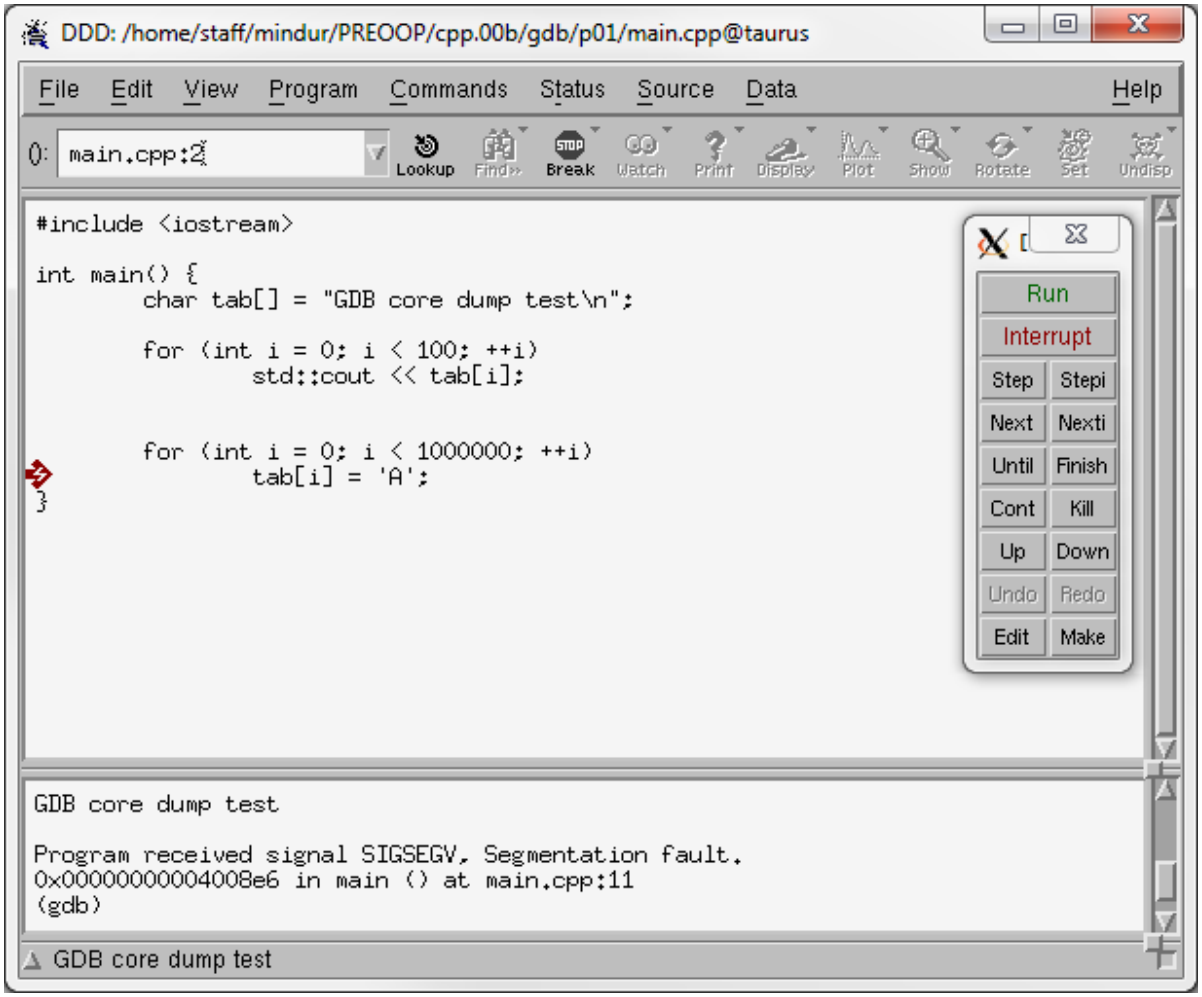
GDB – GNU debbuger

- Jest to standardowy debugger w systemach linuxowych
- Współpracuje z wieloma językami
  - C, C++, Pascal, Fortran, Java
- Jest dostępny na wielu platformach sprzętowych
  - Podobnie jak gcc
- Umożliwia zdalne debugowanie
- Domyślnie pracuje w środowisku tekstowy z interaktywną konsolą
  - Istnieje sporo „nakładek" graficznych na gdb
    - DDD, Xxgdb, Kdevelop, Qt Creator, Eclipse …
- Gdb umożliwia również pisanie skryptów
  - Z wykorzystaniem komend

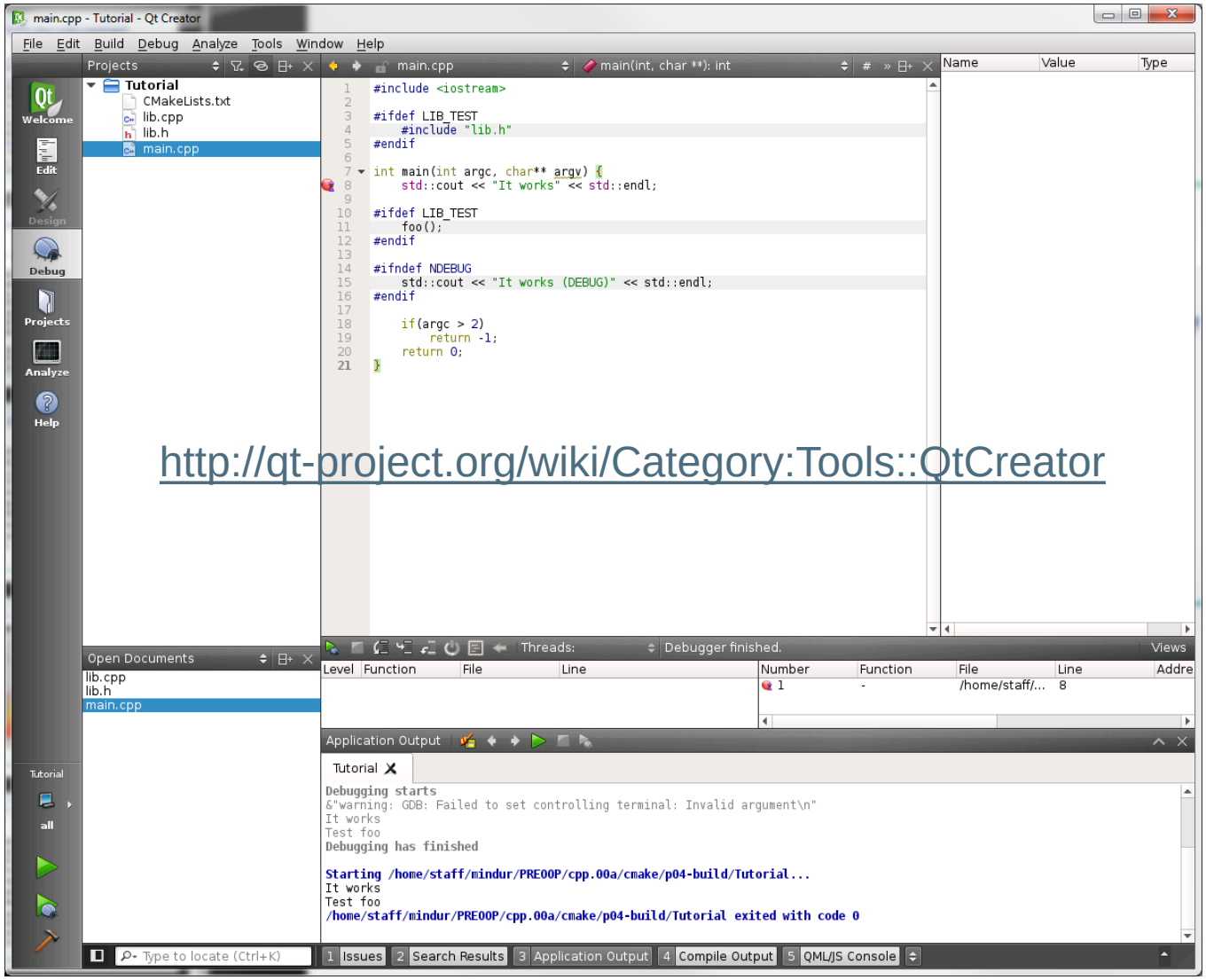
Najczęściej używane komendy GDB

- b main - Puts a breakpoint at the beginning of the program
- b - Puts a breakpoint at the current line
- b N - Puts a breakpoint at line N
- b +N - Puts a breakpoint N lines down from the current line
- b fn - Puts a breakpoint at the beginning of function "fn"
- d N - Deletes breakpoint number N
- info break - list breakpoints
- r - Runs the program until a breakpoint or error
- c - Continues running the program until the next breakpoint or error
- f - Runs until the current function is finished
- s - Runs the next line of the program
- s N - Runs the next N lines of the program
- n - Like s, but it does not step into functions
- u N - Runs until you get N lines in front of the current line
- p var - Prints the current value of the variable "var"
- bt - Prints a stack trace
- q - Quits gdb
- Przykład 1 i 2

DDD – Data Display Debugger



Qtcreator



http://qt-project.org/wiki/Category:Tools::QtCreator

Valgrind

- Valgrind to cały zestaw narzędzi do profilowania i debugowania aplikacji wraz ze wszystkimi podłączonymi bibliotekami dynamicznymi
- W skład pakietu wchodzi kilka narzędzi
  - Do śledzenia „segmentation fault" z minimalnym narzutem
  - Do „walki" z problemami z pamięcią - memcheck
  - Z problemami związanymi z wielowątkowością - helgrind
  - Do profilowania cache'u – cachegrind
  - Do analizy wywołań funkcji – callgrind
  - I kilka eksperymentalnych narzędzi
- Dostępny jest na platformy UNIX-owe, ARM/Android

Valgrind - memcheck

- Śledzi zmiany w alokacji i de-alokacji pamięci
  - Zamienia standardową bibliotekę alokującą pamięć znaną z języka C
  - Umożliwia wykrycie nawet malutkich problemów (nadpisanie/odczytania elementu tablicy o 1 za dużo)
  - Pozwala także na wykrycie
    - Wykorzystania niezainicjalizowanej pamięci
    - Odczyt/zapis z pamięci już zwolnionej
    - Odczyt/zapis poza granicami zaalokowanej pamięci
    - No i oczywiście raportuje wycieki pamięci
- Oczywiście wszystkie te udogodnienia są zrealizowane kosztem utraty wydajności
  - Programy uruchamiane w tym środowisku działają nawet do 30 razy wolniej
  - Dlatego wykorzystuje się go głównie w ramach testów a nie do ciągłej pracy z gotowymi projektami

Memcheck - ograniczenia

- Generalnie występują problemy z wykrywaniem błędów w ramach zmiennych statycznych oraz stosu

```
int Static[5];

int func(void)
{
  int Stack[5];

  Static[5] = 0;  /* Error - Static[0] to Static[4]
exist, Static[5] is out of bounds */
  Stack [5] = 0;  /* Error - Stack[0] to  Stack[4]
exist,  Stack[5] is out of bounds */

  return 0;
}
```

- Jednakże powstało eksperymentalne narzędzie - exp-sgcheck, które umożliwia wykrywanie tego typu błędów
- Przykład 1 i 2

Wyniki testu wycieku pamięci pamięci

- "definitely lost" – zdecydowanie na pewno wyciek pamięci - należy poprawić kod
- "indirectly lost" – stracone pośrednio, np. przez utratę wskaźnika do korzenia drzewa przez co wszystkie dzieci też są utracone
  - Naprawienie definitely lost powinno załatwić sprawę
- "possibly lost" – oznacza że jest wyciek pamięci, chyba że robimy jakieś dziwne rzeczy z wskaźnikami do środka zaalokowanego bloku
- "still reachable" – na ogół znaczy że jest OK, ale jakieś bloki nie zostały zwolnione
- "suppressed" – ignoruję błędy np. pochodzące z bibliotek systemowych

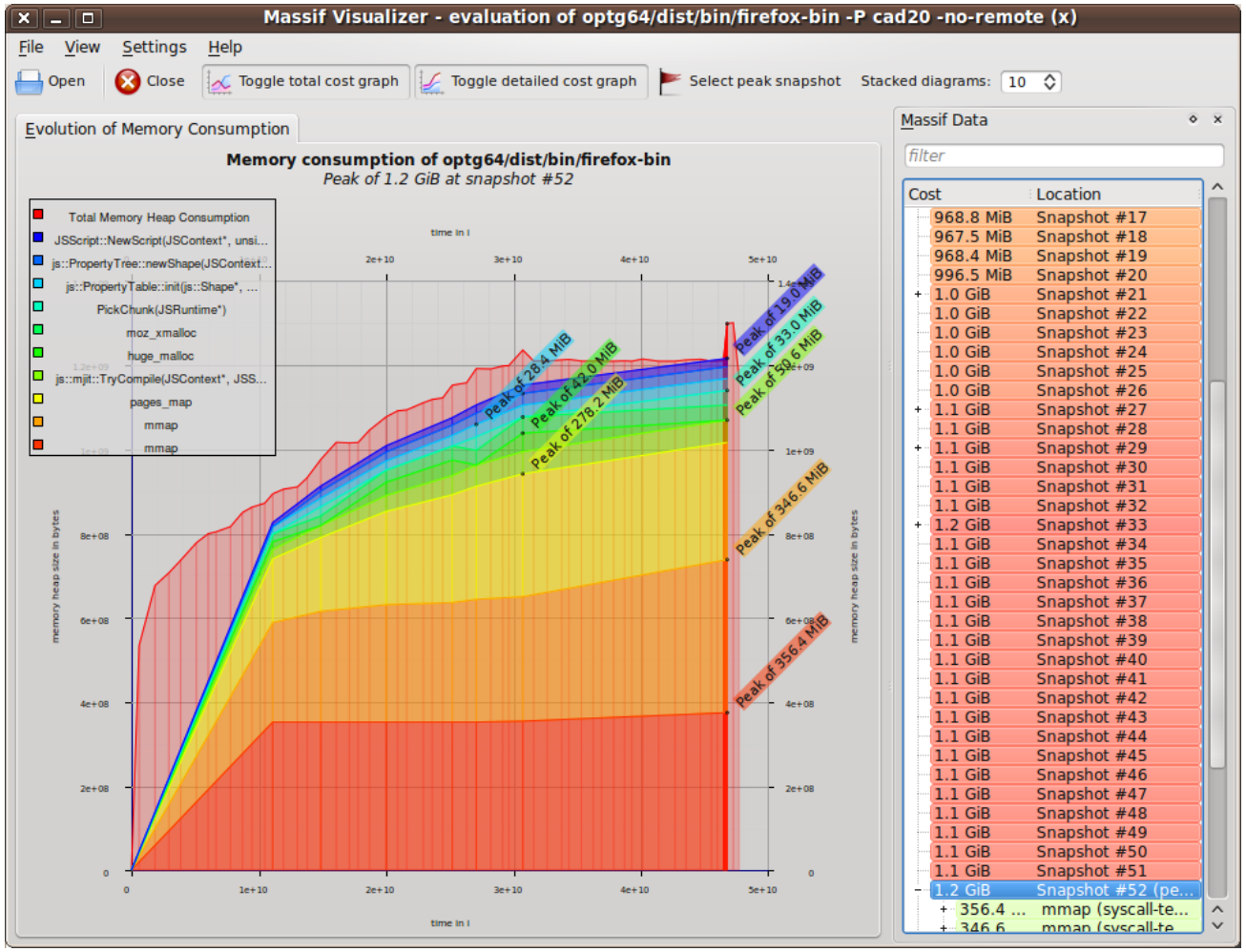
Dodatkowe wyniki w testach pamięci

- Wykrywanie użycia niezainicjalizowanych zmiennych na stosie
- Niepoprawne/podwójne zwalnianie pamięci
- Użycie nieodpowiedniej funkcji do de-alokacji pamięci
  - free/delete
  - delete/delete[]
- Nakładanie się bloków źródłowych i docelowych
  - Np. kopiowaniu
- Raport memcheck zależny jest także od wielu opcji programu
  - --leak-check=<no|summary|yes|full> [default: summary]
  - --leak-resolution=<low|med|high> [default: high]
  - --track-origins=<yes|no> [default: no]

Massif – profilowanie sterty

- Wykonuje pomiar ile i w jaki sposób program wykorzystuje stertę
  - Dotyczy to zarówno bajtów przydzielonych i użytecznych dla programu jak i pomocniczych
  - Pozwala również określać rozmiar stosu
- Bardzo przydatne w celu zmniejszenia pamięcio-żerności program
  - Lepsza optymalizacja cache procesu
  - Mniejsza potrzeba użycia pliku wymiany
- Przykład 4

Valgrind - wizualizacje



Massif Visualizer

Valgrind - uwagi

- Bardzo przydatne aczkolwiek trudne narzędzie
  - Zdecydowanie warto poświęcić czas na naukę
  - W szczególności jest multum opcji do jego konfiguracji
  - Należy również używać z rozwagą i zrozumieniem
- Pozwala na dogłębną analizę wykonywania programów
- Umożliwia szybkie znalezienie wycieków/niepoprawności w obsłudze pamięci
- Pozwala na optymalizację działania oprogramowania oraz testowanie go w warunkach przez nas zdefiniowanych
  - Wiele możliwości valgrinda nawet nie zostało wspomnianych

clang

- sanitize=address
- sanitize=leak

Błędy oprogramowania (spektakularne) - wikipedia

- The European Space Agency's Ariane 5 Flight 501 was destroyed 40 seconds after takeoff (June 4, 1996)
  - The US$1 billion prototype rocket self-destructed due to a bug in the on-board guidance software
- NASA Mars Polar Lander was destroyed because its flight software mistook vibrations due to atmospheric turbulence for evidence that the vehicle had landed and shut off the engines 40 meters from the Martian surface (December 3, 1999)
- A bug in the code controlling the Therac-25 radiation therapy machine was directly responsible for at least five patient deaths in the 1980s
- The 2003 North America blackout was triggered by a local outage that went undetected due to a race condition in General Electric Energy's XA/21 monitoring software
- And many more