Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Prof. T. Ludwig, M. Kuhn, J. Kunkel

Übungsblatt 3 zur Vorlesung
Hochleistungsrechnen
im SoSe 2010
Abgabe: 03.05.2010

Debugging von C-Programmen (300 Punkte)

Um später effizient programmieren zu können, wollen wir uns ein wenig mit der Fehlersuche in (parallelen) Programmen beschäftigen. Hierzu schauen wir uns den GNU Debugger `gdb` und den Speicherprüfer `memcheck` von der `valgrind`-Tool-Suite an. Diese können sowohl für sequentielle als auch für parallele Programme genutzt werden. In C ist die Speicherallokation sehr fehlerträchtig, `valgrind` hilft hierbei typische Fehler aufzuspüren. Eine Liste von Links zu diesen beiden Programmen und wie diese auf dem Cluster verwendet werden können finden Sie unter `http://ludwig9.informatik.uni-heidelberg.de/wiki/index.php/Software_Anleitungen`.
Auf der Materialienseite befindet sich das `.tar.gz`-Archiv `02-gdb-valgrind.tar.gz`, in dem Zusatzmaterialien für die folgenden Aufgaben enthalten sind. Entpacken Sie dieses in ihrem Home-Verzeichnis.

Erste Schritte

Im Verzeichnis `simple` ist ein primitives Programm enthalten, welches mit `make` kompiliert werden kann. Dieses Programm dient nur dazu, dass Sie sich ein wenig mit `gdb` und `valgrind` beschäftigen. Dieses Programm enthält lediglich vier Funktionen, welche jeweils einen Zeiger auf eine Zahl oder ein Array mit einer Zahl enthalten und gibt diese dann in der `main()`-Funktion aus. Leider enthält dieses Programm diverse Fehler.

- Führen Sie folgende kleineren Tests durch, um `gdb` kennen zu lernen. Dokumentieren Sie die genutzten Eingabebefehle und die Ausgabe von `gdb` in einer Textdatei:
  - Platzieren Sie einen Breakpoint in der Funktion `mistake1()`, starten Sie das Programm, drucken Sie den Wert von `buf` und `buf[2]` aus. Gehen Sie zur nächsten Zeile und drucken Sie beide Werte wieder aus. Von welchem Typ ist `buf`?
  - Platzieren Sie einen Breakpoint in der Funktion `mistake2()`, setzen Sie den Programmlauf fort, welchen Typ hat `buf`?
  - Setzen Sie den Programmlauf fort, welcher Text wird nun ausgegeben? Lassen Sie sich den Code um diese Stelle herum ausgeben. Welche Frames sind auf dem Stack? Wechseln Sie zu Frame 1. Drucken Sie den Inhalt von `p` aus.
  - Rufen Sie im `gdb` die Funktion `mistake4()` auf (schauen Sie nach, wie man in `gdb` Funktionen direkt aufrufen kann).
- Modifizieren Sie das Programm zunächst so, dass es nicht mehr abstürzt. Versuchen Sie die Modifikationen möglichst gering zu halten. Verwenden Sie zunächst `gdb` um die Fehlerstellen aufzuspüren.
- Nun läuft das Programm, leider enthält es jedoch noch weitere Speicherfehler, die je nach Umgebung (mehr oder weniger zufällig) auftreten können. Modifizieren Sie das Programm unter Zuhilfenahme von `valgrinds memcheck` so, dass jede Methode Speicher korrekt reserviert und dass am Ende der Programmlaufzeit der Speicher korrekt freigegeben wird.

Dokumentieren Sie die Programmierfehler, die zu den Abstürzen und Speicherfehlern führen. Notieren Sie hierfür für jeden vorhandenen Fehler die Code-Zeile(n), welche fehlerhaft sind und den genauen Grund der Ursache (z. B. Speicher mehrfach freigegeben).

Abgabe

Eine Textdatei mit den gefundenen Ein- und Ausgaben von gdb `gdb-ausgabe.txt`, eine Textdatei `simple-error.txt` mit Fehlerbeschreibung (Ursache, Code-Zeile(n)). Modifizierter Quelltext.

Debugging einer komplexeren Anwendung

Im Verzeichnis `broken-PDE` ist ein numerisches Programm zum Lösen von Differentialgleichungen. Grundsätzlich ist das Programm vom Ablauf korrekt, jedoch haben sich durch Unachtsamkeit einige Flüchtigkeitsfehler bei der Speichernutzung eingeschlichen. Um das Programm zu korrigieren, müssen Sie nicht genau verstehen was berechnet wird, u. U. wollen Sie jedoch mit einem Debugger die Aufrufe ein wenig verfolgen. Korrigieren Sie alle Speicherfehler im Programm. Modifizieren Sie hierbei den Code so wenig wie nötig.

Hinweise:

- Eine Anpassung des Makefile für das Debugging ist notwendig.
- Das Programm sollte zum Testen wie folgt aufgerufen werden: `./partdiff-seq 1 1 100 1 2 5`

Abgabe

Eine Textdatei `pde-error.txt` mit Fehlerbeschreibung (Ursache, Code-Zeile(n)). Modifizierter Quelltext (geben Sie das ganze Verzeichnis `broken-PDE` mit ihren Änderungen ab). Schicken Sie ihre Lösungen als `tar`-Archiv an `michael.kuhn@informatik.uni-hamburg.de`.
Denken Sie daran, das korrekt benannte Verzeichnis in das `tar`-Archiv zu packen (siehe Übungsblatt 0)!
Am einfachsten ist es mit `make clean` alle Binärdaten zu entfernen, das `gdb-valgrind`-Verzeichnis umzubenennen und alles wieder in ein `tar`-Archiv zu packen.

Rückmeldung (+ 5-10 Punkte)

Bearbeitungszeit			
Schwierigkeit	o zu leicht	o genau richtig	o zu schwer
Lehrreich	o wenig	o etwas	o sehr
Verständlichkeit	o großteils unklar	o teilweise unklar	o verständlich
Kommentar			