第 10 回 Unix ゼミ
C プログラム(デバッグ編)
演習

高木 空

2024 年 07 月 01 日

目次

1 演習 1 (GDB) 2
2 演習 2 (LLDB) 2
3 演習 3 (Perf) 3
4 演習 1 回答 (GDB) 3
5 演習 2 回答 (LLDB) 3
6 演習 3 回答 (perf) 4

1 演習 1 (GDB)

この演習は `ex1-2/`に必要なリソースが入っています。`ex1-2/`に移動してください。

以下のことを GDB で実行してください。

1. デバッグ情報を付与して `test.c` をコンパイルしてください。
2. 1. で作成した実行ファイルを読み込んで GDB を起動してください。
3. 以下の場所にブレークポイントを設置してください。
   - 63 行目
   - 35 行目 (`i<=j` の条件付き)
   - 関数 `qsort` (`low>high` の条件付き)
4. プログラムを起動してください。(63 行目で停止したことを確認)
5. 変数 `data` を長さ 12 の配列として表示してください。
6. 関数の中にステップ実行で入ってください。(関数の先頭で停止したことを確認)
7. 変数 `data` を長さ 12 の配列として表示してください。
8. 次の停止場所まで進んでください。(35 行目で停止したことを確認)
9. 変数 `i` と `j` の値を表示してください。(5, 11 となっていることを確認)
10. 次の停止場所まで進んでください。
11. バックトレースを確認してください。(`main` 関数が `#2` であることを確認)
12. 35 行目に設置したブレークポイントを削除してください。
13. 次の停止場所まで進んでください。(`low=0, high=-1` の `qsort` 呼び出し内であることを確認)
14. ブレークポイントを全て削除してください。
15. 一番最初の `qsort` 関数が終了するまで実行してください。
16. 変数 `data` を長さ 12 の配列として表示してください。(ソートされていることを確認)
17. 最後まで実行して GDB を終了してください。

回答 Section 4

2 演習 2 (LLDB)

この演習は `ex1-2/`に必要なリソースが入っています。`ex1-2/`に移動してください。

演習 1 と全く同じことを LLDB で行ってください。

回答 Section 5

3 演習 3 (Perf)

この演習は ex3/に必要なリソースが入っています。ex3/に移動してください。

test.c には 2 つの行列の和を求めるプログラムが入っています。 アクセスパターンによるキャッシュミスの増減を perf を使って確認しましょう。

先ずは mut_add_a のコメントアウトを消してコンパイルしてください。

次に perf stat でイベント L1-dcache-load-misses と L1-dcache-loads を集計してください。

次に mut_add_b に切り替えて再度コンパイルし、同様にプロファイルを行ってください。

そして L1 キャッシュミスと実行時間を確認してください。

また、いづれかの方で perf record を実行し、perf report を試してみてください。

4 演習 1 回答 (GDB)

```
$ gcc -g test.c -o a.out
$ gdb ./a.out
(gdb) break 63
(gdb) break 35 if i<=j
(gdb) break qsort if low>high
(gdb) run
(gdb) print data
(gdb) step
(gdb) print *data@12
(gdb) continue
(gdb) print i
(gdb) print j
(gdb) backtrace
(gdb) delete 2 (2 かどうかは設置順に依る)
(gdb) continue
(gdb) delete
(gdb) frame 11
(gdb) finish
(gdb) print data
(gdb) continue
(gdb) quit
```

5 演習 2 回答 (LLDB)

```
$ gcc -g test.c -o a.out
$ gdb ./a.out
(lldb) breakpoint set -l 63
(lldb) breakpoint set -l 35 -c i<=j
(lldb) breakpoint set -n qsort -c low>high
(lldb) process launch
(lldb) frame variable data
(lldb) thread step-in
(lldb) frame variable -Z 12 data
```

```
(lldb) thread continue
(lldb) frame variable i j
(lldb) thread continue
(lldb) thread backtrace
(lldb) breakpoint delete 2 (2 かどうかは設置順に依る)
(lldb) thread continue
(lldb) breakpoint delete
(lldb) frame select 11
(lldb) thread step-out
(lldb) frame variable data
(lldb) thread continue
(lldb) quit
```

6 演習 3 回答 (perf)

```
$ perf stat -e L1-dcache-load-misses,L1-dcache-loads ./a.out
```