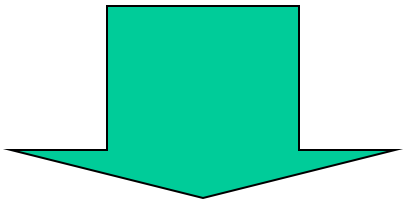
デバッガ dbx の使い方

デバッガとは

- プログラムのバグを追跡し，特定し，排除する作業を援助するツール
- プログラムの動的な性質を明らかにするツール

デバッグの必要性

- プログラムは正しくコンパイルできるが，実行結果が正しくない



- デバッガを使い，正しく動作しない原因を探し，プログラムの変更を行う

dbx の起動

- dbx *program_name*
  - *program_name*のプログラムのデバッグを行う
  - 注意：デバッグを行う場合，－g オプションを付けてコンパイルを行う
  - 例：CC –g –o sample sample.cc

    dbx ./sample

基本的なコマンド(1)

- run
  - プログラムを実行する
- runargs <args>
  - 実行時の引数を設定する
- runargs
  - 現在の引数をクリアする
- stop at <line>
  - 指定した行番号<line>にブレークポイントを設定する
- stop in <func>
  - 指定した関数<func>をブレークポイントに設定する

基本的なコマンド(2)

- cont
  - ブレークポイントまでプログラムを実行し，停止する
- rerun
  - プログラムを最初から実行する
- next
  - プログラム停止状態で次の行を実行する
- next <n>
  - nextを<n>回繰り返す

基本的なコマンド(3)

- step
  - プログラム停止状態で次の行を実行する
  - 次の行が関数なら，その関数に入る
- step <n>
  - stepを<n>回繰り返す
- dump
  - 現在の関数の全てのローカル変数を表示する
- print <exp>
  - <exp>の値を表示する

基本的なコマンド(4)

- up
  - 1つ上の関数に移動
- up <n>
  - <n>個上の関数に移動
- down
  - 1つ下の関数に移動
- down <n>
  - <n>個下の関数に移動

dbx の使い方(1)

- コンパイル
  - デバッグを行うときに，－gオプションを付けてコンパイルを行う
- デバッガの起動
  - dbx ./TPCBenchmark
- 実行ファイルと引数の確認
  - (dbx) debug
  - Debugging: TPCBenchmark
    - Debugging: プログラム名　第1引数　第2引数　･･･　というように表示される
    - この例では，引数は設定されていない
- 引数の設定
  - (dbx) runargs kei ../tpc-c/tpc.config.3000 0 1 3000 1
    - runargs　第1引数　第2引数　･･･というように入力する

dbx の使い方(2)

- 実行ファイルと引数の確認
  - (dbx) debug
  - Debuggin: TPCBenchmark kei ../tpc-c/tpc.config.3000 0 1 3000 1
    - 引数が設定されていることがわかる
- ブレークポイントの設定
  - (dbx) stop in main
  - (2) stop in main
    - main関数に入ったところでプログラムが停止するように設定
- プログラムの実行
  - (dbx) run
  - stopped in main at line 33 in file “TPCBenchmark.cc”
  - 33 if(argc != 7){
    - プログラムを実行し，最初のブレークポイントであるmain関数に入ったところで停止する

dbx の使い方(3)

- ローカル変数の表示
  - (dbx) dump
  - argv = 0xffffffff7ffff538
  - rb = 0
  - mpl = 0
  - set_timer_request_1_arg = 2147478097
  - open_flag = 0
  - new_order_tp = -1
  - part_name = ""
  - payment_tp = 0
  - heap_reg_arg = STRUCT
  - result = (nil)
  - host = (nil)
  - arg = STRUCT

dbx の使い方(4)

- argc = 7
- stock_level_tp = 0
- order_status_tp = 0
- delivery_tp = 0
- clnt = (nil)
- result_info = 0x1000551cc
  - main関数で使われている変数名と値が表示される

- 指定した変数の表示
  - (dbx) print argv[1]
  - argv[1] = 0xffffffff7ffff859 “kei”
    - argv[1]の値を表示
    - 指定した変数がポインタなので，ポインタの中身も表示される

dbx の使い方(5)

- プログラムの表示
  - (dbx) list
  - 33 if(argc != 7){
  - 34 fprintf(stderr, "Usage: %s host_name configure_name database_open period transaction_number mpl¥n", argv[0]);
  - 35 exit(0);
  - 36 }
  - 37 char* host = argv[1];
  - 38 // Create RPC Client
  - 39 CLIENT* clnt;
  - 40 clnt = clnt_create(host, SERVERPROG, SERVERVERS, "tcp");
  - 41 if (clnt == (CLIENT *) NULL) {
  - 42 clnt_pcreateerror(host);
    - 現在停止している場所からプログラムが10行表示される

dbx の使い方(6)

- 次の行を実行
  - (dbx) next
  - stopped in main at line 37 in file “TPCBenchmark.cc”
  - 37 char* host = argv[1]
    - プログラムを1行実行して，停止する
- hostに値が格納されることを確認する
  - (dbx) print host
  - host = (nil)
  - (dbx) next
  - stopped in main at line 40 in file “TPCBenchmark.cc”
  - 40 clnt = clnt_create(host, SERVERPROG, SERVERVERS, "tcp");
  - (dbx) print host
  - host = 0xffffffff7ffff859 “kei”
    - hostの値が(nil)から0xffffffff7ffff859に変わっている

dbx の使い方(7)

- 指定した行をブレークポイントに設定する
  - (dbx) stop at 53
  - (3) stop at “TPCBenchmark.cc”:53
    - TPCBenchmark.ccの53行目がブレークポイントに設定される
- ブレークポイントまで実行する
  - (dbx) cont
  - stopped in main at line 53 in file “TPCBenchmark.cc”
  - 53 LoadEnv(argv[2]);
    - 53行目をブレークポイントに設定していたので，53行目で停止
- ブレークポイントの確認
  - (dbx) status
  - (2) stop in main
  - *(3) stop at “TPCBenchmark.cc”:53
    - 現在設定されているブレークポイントが表示される
    - *がついている場所が現在停止している場所

dbx の使い方(8)

- ブレークポイントの削除
  - (dbx) clear “TPCBenchmark.cc”:53
  - cleared (3) stop at “TPCBenchmark.cc”:53
  - (dbx) status
  - (2) stop in main
    - ブレークポイントが削除されたことが確認できる
- 関数の中に入る
  - (dbx) step
  - stopped in LoadEnv at line 112 in file “TPCBenchLib.cc”
  - 112 SetParams(database_config_name);
    - 関数(LoadEnv)の中に入り，停止

dbx の使い方(9)

- (dbx) step
- stopped in SetParams at line 62 in file “SetParams.cc”
- 62 tpcdbname = new char[40];
  - 関数(SetParams)の中に入り，停止

- プログラムの停止している場所を表示
  - (dbx) where
  - current thread: t@1
  - =>[1] SetParams(configFileName = 0xffffffff7ffff85d "../tpc-c/tpc.config.3000"), line 62 in "SetParams.cc"
  - [2] LoadEnv(database_config_name = 0xffffffff7ffff85d "../tpc-c/tpc.config.3000"), line 112 in "TPCBenchLib.cc"
  - [3] main(argc = 7, argv = 0xffffffff7ffff538), line 53 in "TPCBenchmark.cc“
    - 関数名(引数リスト), line 行番号 in “ファイル名”と表示される
    - 矢印のついてる場所[1]が現在停止している場所

dbx の使い方(10)

- 現在の関数(SetParams)から抜ける
  - (dbx) stop at “TPCBenchLib.cc”:113
  - (4) stop at “TPCBenchLib.cc”:113
    - SetParamsを呼び出したプログラムの次の行をブレークポイントに設定
  - (dbx) cont
  - stopped in LoadEnv at line 113 in file “TPCBenchLib.cc”
  - 113 }
    - SetParamsが実行され，プログラムが停止
- 変数のタイプを調べる
  - (dbx) whatis database_config_name
  - char *database_config_name;
    - database_config_nameはchar型のポインタである

dbx の使い方(11)

- プログラムを最後まで実行する
  - (dbx) cont
  - MQTH = 701 (701, 649, 74, 65, 69)
  - Rollback = 0
  - execution completed, exit code is 0
    - プログラムの実行結果が出力され，正しく終了したことを示す