SCO Debugger – Tips and Tricks

Ron Record
rr@sco.com

John Wolfe
jlw@sco.com

Agenda

- Truss
  - trace system calls & signals
- Debug
  - Command line interface
  - Graphical user interface
- Memtool
  - catching dynamic memory errors
- SVR5 MALLOC_CHECKS

Debugging

truss

Truss – trace system calls & signals

- One or many processes
  - Optionally follow forked process(es)
  - Optionally indicate LWP id of threaded process(es)
- Asserts control (monitors) process through /proc file system
  - Supports both SVR5 and OSR5 ABI processes
- Selectively display or suppress:
  - System Calls, Signals, Machine Faults
  - Complete I/O by file descriptor
- Display system call arguments

Truss – command format

truss [-flcaein] [-[tvx] [!] syscall . . .]
[-s [!] signal . . .] [-m [!] fault . . .]
[-[rw] [!] fd . . .] [-o outfile] command | -p pid

- **-f : follow child processes**
- **-l : display LWP on threaded programs**
- **-a : display string arguments to “exec”**
- **-e : display the ENVIRONMENT passed to “exec”**
- **Defaults**
  - **-tall -v!all -x!all -sall -mall -m!fltpage -r!all -w!all**

Debugging with debug

SCO

OpenServer 6 Debugger - debug

- Graphical user interface
  - user configurable screen layouts
  - command aliases
- Command line interface
  - powerful, shell-like command language
  - command history, command aliases
- Strong C and C++ symbolic debugging
  - step through inline functions, header code, exceptions
- Controls multi-process, multi-threaded apps
  - follow forks in both parent and children processes
- Understands ELF/COFF, DWARF I/II executables

debug - How to get started

- **debug** – man page
- **help** command in the debugger
  - **help** – lists available commands and topics
  - **help <cmd-topic>** - format and details about specific command or topic
- Use the on-line/locally installed debugger doc. “Debugging and analyzing C & C++ Programs”
  - Command line and GUI
    - Tutorials, explanations, and tips

debug - Command Format

1. **debug [com_opts] [[-p] [-f all|none|procs] \ [-r] [-l start_loc] cmd_line]**

2. **debug [com_opts] [-p] [-m path] -c core_file \ [object]**

   debug [com_opts] [-p] [-m path] core_file

   com_opts: [-V] [-i c|x] [-X opt] [-d defaults] [-s path] [-Y[a|g],dir]

- Debugger variables
  - Begins with '%'
  - **Execution state**
  - **Debugger attributes**
    - %program %proc %thread %log
      %func %file %line %frame
      %lastevent %thisevent %eh_object cpu_registers
      %list_file %list_line
    - %db_lang and %lang – "C" or "C++"
    - %follow – control following of child processes
      - "all" or "procs" versus "none"
    - %mode – current line editing mode
      - "vi" or "emacs"
      - Initial setting by VISUAL or EDITOR environment setting

- Debugger attributes – continued
  - **%num_lines – default lines printed for list and dis commands**
  - **%num_bytes – default number of bytes displayed by dump**
  - **%wait - synchronous or asynchronous command execution**
    - **Synch. - "foreground", 1 or "yes"**
    - **Asynch. - "background", 0 or "no"**
  - **%thread_change - control behavior on state changes**
    - **“stop”**
    - **“announce”**
    - **“ignore”**
  - **%global_path - debugger's global search path**
    - **Supplemental source search path**

- **User defined, debugger maintained variables**
  - **Begin with a dollar sign - '$'**
  - **Imports shell environment variables at start up**
  - **Create with set command**
  - **type is “string”**
    - **Converted to integer as needed - strtol()**

- **proclist – comma-separated list of procnames**
- **procnames**
  - **“all” - all controlled processes and threads**
  - **user / debugger generated program name**
  - **debugger process id - p<n>**
  - **debugger thread id - p<n>.<n>**
  - **system process id - integer**
  - **“current” %program, %process or %thread**
  - **user debugger variable with integer process id**

- **Location**
  - [thread_id@][object@]**address**[+-constant]
  - [thread_id@][object@][source_file@][header_file@]**line**
  - [thread_id@][object@][source_file@][header_file@]
    function[+-constant]
- Qualified identifier
  - [thread_id@][source_file@][function@][line_number@]
    identifier
  - [thread_id@][source_file@][header_file@]**identifier**
  - [thread_id@]**frame_number@identifier**
  - [thread_id@]object_name@[source_file@][header_file@]
    identifier

- **Expression**
  - **Combination of:**
    - **Variables (program, debugger, user debugger)**
    - **Functions**
    - **Qualified names**
  - **Syntax of “current” language**
  - **Enclose in parens, square brackets or curly braces**
    - **Begins with '-'**
    - **Contains:**
      - **>, >>, |, ||, &&, #, comma, semi-colon, newline**

- Creating a debug session
  - create command – create new process(es)
    - **create [-dpr] [-f all|none|procs] [-l start_loc] [cmd_line]**
  - **grab command – grab a running process or corefile**
    - **grab [-f all|none|procs] [-l load_file] process_spec**
    - **grab [-p] [-m path] -c corefile [objectfile]**
    - **grab [-p] [-m path] corefile**

- **Process Execution**
  - **run [-p proclist] [-bfr] [-u location]**
  - **step [-p proclist] [-iobfq] [-c count]**
    - **next predefined alias for "step -o"**
  - **release [-s] [-p proclist]**
  - **halt [-p proclist]**

- Stop events
  - Break points – function, statement, instruction address
  - Watch point – value in memory changes
    - ***lvalue**
  - Expression – logical expression is true-
    - **(expr)**
- Signals - d**efault: monitors every signal**
- C++ exceptions - default: every **throw and catch**
- System calls
- On Stop event

- Creating **stop events**
  - **stop [-p proc_list] [-c count] stop_expr [command]**
  - **stop [-p proc_list]**
  - **aliased as b**
- **Managing C++ exception events**
  - **exception -d [-i] {throw|catch}**
    - **set default action henceforth**
  - **exception [-p proclist] [-iq] {throw|catch} [type] [command]]**
  - **%eh_object – current exception object**

- **Managing signal actions**
  - **signal -d [-i] [signal ...]**
    - **set default action henceforth**
  - **signal [-p proclist] [-iq] [signal ... [command]]**
  - **signal [-p proclist] -m**
    - **displays signal mask**
  - **cancel [-p proclist] [signal ...]**
    - **cancel delivery of pending signal(s) to designated proclist**
  - **kill [-p proclist] [signal]**
    - **send signal to designated proclist**

- Tracing of system calls
  - **syscall [-p proclist] [[-eqx] [-c count] call ... [command]]**
  - **use system call name or number**
    - **help sysnames**
  - **-e on entry**
  - **-x on exit**
- **On Stop Events**
  - **onstop [-p proclist] [command]**
  - **NOTE: single stepping constitutes a stop**

debug - Creating/Managing Events

- **events [-p proclist] [event_num ...]**
  - **lists all or the designated events**
- **{delete | disable | enable} event_num ...**
  - **delete, disable or enable the specified events**
- **enable -a [-p proclist] [event_type]**
  - **delete, disable or enable ALL events of the specified event type**
- **change event_num [-p proclist] [-evqx] [-c count] [throw|catch] [stop_expr|call...|signal...| exception_type] [{commands}]**

debug - Displaying Data & Process Information

- **ps [-p proclist]**
  - **list status of controlled threads and processes**
  - *** marks current thread or process**
- **stack [-p proclist] [-f frame] [-c count] [-a address] [-s stack]**
  - **display function call backtrace**
  - *** marks the current frame**
  - **Aliased as t**
- **map [-p proclist]**
  - **display virtual address map**

- List source lines
  - **list [-p proclist] [-c count]**
    - **list from “current” location**
    - **list next set of lines if repeated**
  - **list [-p proclist] [-c count] qualified_src_location**
    - **function name or source file and line number**
  - **list [-p proclist] [-c count] /regexp/**
    - **list from the next line which matches the regexp**
  - **list [-p proclist] [-c count] ?regexp?**
    - **Search backwards for the line that matches the regexp**

- **Display symbol names, values and types**
  - **symbols [-p proclist] [-o object] [-n filename] [-dfgltuv] [pattern]**
- **Print value of an expression**
  - **print [-p proclist] [-f format] [-v] expr, ...**
    - **expr evaluated in “current” language – C or C++**
    - **format is format string acceptable to C printf()**
- **Display the type of an expression**
  - **whatis [-p proclist] expr**

debug - Displaying Data & Process Information

- **Display contents of memory**
  - **dump [-p proclist] [-c byte_count] [-b] expression**
- **Disassemble machine instructions**
  - **dis [-p proclist] [-c instr_count] [-ns] [location] [end_location]**
- **Display machine registers – general, FP and MMX**
  - **regs [-p proclist]**

debug - Personal Configuration

- **alias** command
  - define alternate / abbreviated commands
  - use to establish dbx-like or gdb-like commands
  - build complex, repetitive, conditional command sequences
- **$HOME/.debugrc**
  - startup **debug** command script
  - establish *my_former_debugger*-like configuration
  - **debug … –d <alt_startup> …**
    - uses specific alternate startup script instead of default

debug - logon / logoff / script

- **logon <log_file>**
  - logs debug commands entered and output to a file
  - generated output appears as comments
  - capture complete history
  - capture repetitive command sequence
- **logoff**
  - Terminate logging
- **script <file>**
  - reads debug commands from **<file>**

debug - For dbx or gdb users

- **dbx** users
  - Section 3 of the Porting Guide “A Guide to debug for dbx Users”
- **gdb** users
  - command comparisons from May/June 2000 SCO World article
    - Summary is in the on-line handout

Debug – GUI default layout

Debugging: Dynamic Memory

memtool

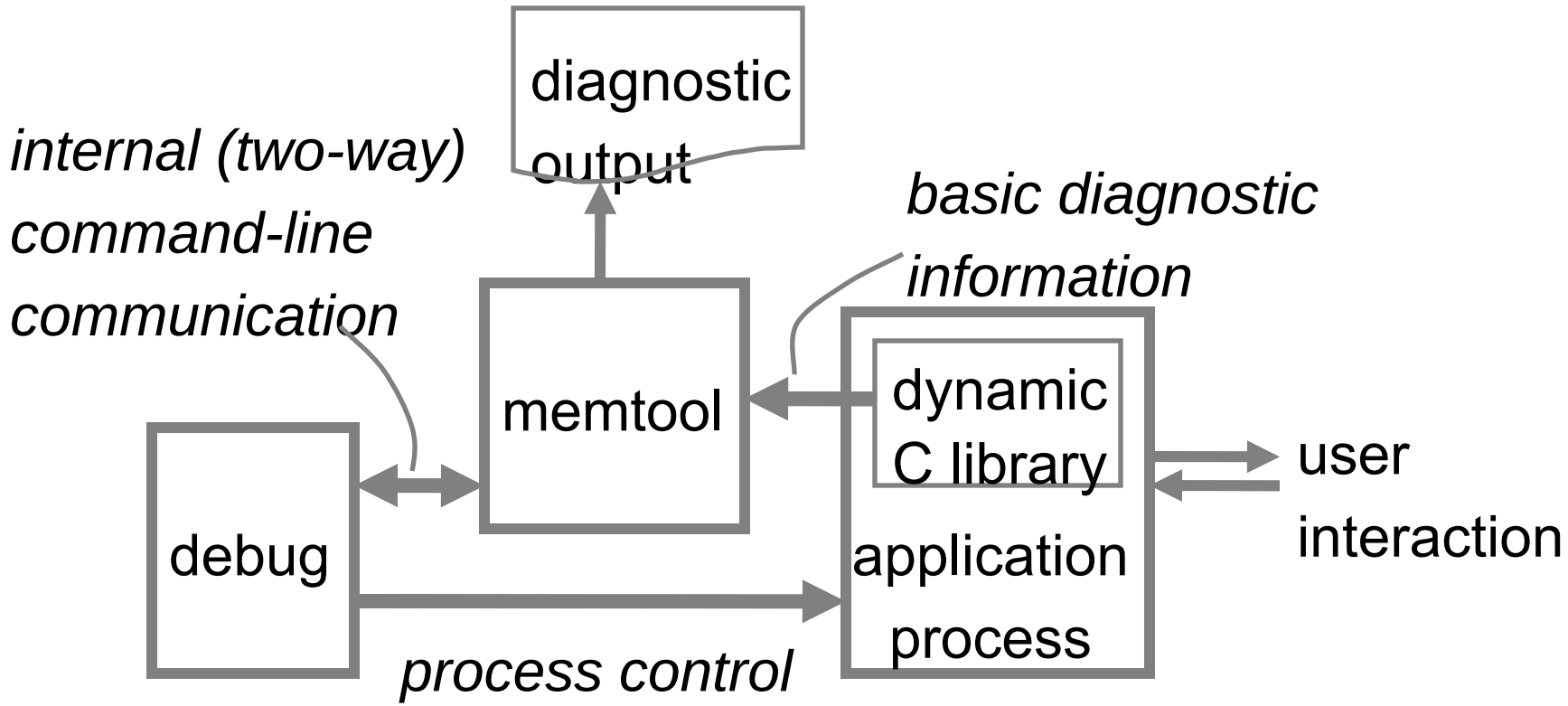
SCO

memtool - Catching Dynamic Memory Errors

[SVR5/UDK ABI only]

- Diagnose dynamic memory allocation errors
  - writing beyond a block of memory
  - using deallocated blocks
  - memory “leaks”
  - bad arguments passed to C **malloc()** or C++ **new()**
- Does not catch general pointer misuses or writing outside local or global arrays
- Runs the application under the hidden control of the debugger and the dynamic C library **malloc** runtime checking

memtool – Under the covers



memtool (cont'd)

- Diagnostics include one to three stack traces
  - when detected
  - when (de)allocated
  - previous use (for **realloc()** or **free()**)
- Erroneously modified block diagnostics include an annotated memory dump snapshot for the block
- Each diagnostic comes with an explanation – short, medium, or long (user selectable)
- Application need not be rebuilt or relinked
  - debugging (**-g** flag) provides much better info

Debugging: Dynamic Memory

SVR5 - MALLOC_CHECKS

SCO

- Environment variable activated memory checking in the SVR5 C runtime
  - No recompilation needed - dynamic libc.so.1
- MALLOC_CHECKS=<number>
  - 1 = basic-fill mode
  - 3 = safe-copy mode - duplicate arena block hdrs.
  - 5 = added-space mode – allocation padded
  - **mallinfo()** - check arena integrity
  - 2, 4, 6 = above with arena check on all malloc calls
  - -1, -5 = high memory edge with electric fence
  - -3, -7 = low memory edge with electric fence

Guidance / Assistance

OpenServer 6 Support Resources

- Porting Guide:
  - http://www.sco.com/support/docs/openserver/600/porting/osr6portingTOC.html
- Upgrade Guide:
  - http://www.sco.com/support/docs/openserver/600/upgrade/index.html
- Online Documentation and Late News
  - http://www.sco.com/support/docs/openserver/

OpenServer 6 Support Resources

- Support Download Page for OpenServer 6:
  - http://www.sco.com/support/update/download/product.php?pfid=12&prid=20
- Tricks on getting OpenServer 5, UnixWare, SCO Unix and SCO Xenix applications running on SCO OpenServer 6 – Forum 2006
  - http://www.sco.com/2006forum/breakouts/breakout/140_Boland_J_tips-tricks.ppt

OpenServer 6 Support Resources

- SCO “Legend” Mailing List: Public
  - Legend-subscribe@list.sco.com
  - legend@sco.com
- Porting/Migration Alias:
  - osr5to6@sco.com
- Knowledge base:
  - http://wdb1.sco.com/kb/search