Debugging

Interactive debugging

- Based on a source code, statement-level debugger
- Allows to discover values of variables by using their names in the source program, tracing their execution one statement at a time
- The C files are compiled with the `-g` flag in effect
  - Allows the inclusion of extra symbol table information in the binary files
    - Names and locations of all variables
    - Names of all functions and their arguments
    - Data types of all objects declared in the program
    - Path names of the source code files used to compile the program
- `xxgdb` debugger
  - Provides a windows-oriented graphical user interface to `gdb` under the X window system
  - Provides mouse selection for various text commands
  - Allows user to control program execution through breakpoints
  - Consists of the following windows
    - **File window**
      - Displays the full pathname of the file displayed in the source window
      - Also displays the line number of the caret
    - **Source window**
      - Contents of a source file
    - **Message window**
      - Execution status and error messages of `xxgdb`
    - **Command window**
      - List of common `gdb` commands
      - Commands invoked by clicking the left mouse button in the box
    - **Dialog window**
      - Typing interface to `gdb`
    - **Display window**
      - Window to display variable values
    - **Popup windows**
      - Windows for displaying variables
  - Text selection
    - C expression selected by clicking on the left mouse button
    - Based on the resource delimiters to determine the set of characters that delimit a C expression
    - Also possible to select text by holding down the left mouse button and dragging
    - Pressing shift key with left mouse button click displays the value of the variable
  - Scrollbar
    - Press left mouse button to scroll text forward
    - Press right mouse button to scroll text backward
    - Drag the middle mouse button to change the thumb position of the text
  - Command buttons
    - `run`
      - Begin program execution

- `cont`
  - Continue execution from where it stopped
- `next`
  - Execute one source line, without stepping into any function call
- `step`
  - Execute one source line, stepping into a function if the source line contains a call to a function
- `finish`
  - Continue execution until the selected function returns
  - Use current function if none is selected
- `break`
  - Stop program execution at the line or in the function selected
  - Place the caret at the start of source line or on the function name
  - Click the `break` button
  - A stop sign appears next to the source line
- `tbreak`
  - Set a breakpoint enabled for only one stop
  - Same as the `break` button except that the breakpoint is automatically disabled after the first time it is hit
- `delete`
  - Remove the breakpoint on the source line selected, or the breakpoint number selected
- `show brkpts`
  - Show the current breakpoints (both active and inactive)
- `stack`
  - Show a stack trace of functions called
- `up`
  - Move up one level on the call stack
- `down`
  - Move down one level on the call stack
- `print`
  - Print the value of a selected expression
- `print *`
  - Print the value of the object the selected expression is pointing to
- `display`
  - Display the value of a selected expression in the display window, updating it every time execution stops
- `undisplay`
  - Stop displaying the value of the variable in the display window
  - If the selected expression is a constant, it refers to the display number associated with an expression in the display window
- `args`
  - Print the arguments of the selected frame
- `show display`
  - Show the names of currently displayed expressions
- `locals`
  - Print the local variables of the selected frames
- `stack`
  - Print a backtrace of the entire stack