A Short User Manual for dbx

"dbx" is the name of the debugger that is available for use on Silicon and many other Unix systems. This document is intended to list the most common dbx commands and briefly explain how to use them. It is not a complete description of dbx but should be a sufficient reference to get started. For a more detailed description or to find out about other commands and options, execute the command % man dbx or from within dbx use the 'help' command (recommended).

Subjects covered below: let (dbx) be the prompt while in dbx mode:

```
a) Compiling your code for use with the debugger   [ cc with -g flag]
b) Starting the debugger                         [ \%dbx executable         ]
c) Getting help while using the debugger  [ (dbx) help              ]
d) Running your program in the debugger   [ (dbx) run               ]
e) Setting breakpoints in the program     [ (dbx) stop              ]
f) Listing the breakpoints that are set   [ (dbx) status            ]
g) Removing a breakpoint                  [ (dbx) delete            ]
h) Stepping through the program           [ (dbx) step,  (dbx) next ]
i) Jumping ahead to the next breakpoint   [ (dbx) cont              ]
j) Examining the values of variables      [ (dbx) print             ]
k) Listing a portion of the source code   [ (dbx) list              ]
l) Leaving the debugger                   [ (dbx) quit              ]
```

To use dbx to debug your C programs you need to follows the procedure:

1. Compiling your code for use with the debugger: You must use the -g flag when compiling your code. For example:
   % cc -g myfile.c will create an a.out executable suitable for dbx.

2. Starting the debugger: Give the 'dbx' command along with the name of the executable you wish to examine. For example:
   % dbx a.out executes a.out in debigging mode. After this command your process is run in the debug mode (i.e., by dbx) with the prompt (dbx)

3. Getting help while using the debugger: To get help while your process run in the debugging mode simply give the command "(dbx) help" to get a list of keywords and the help syntax. To get details about a particular item, type "(dbx) help keyword" where keyword is an element of the list displayed when you just type help. For example: "(dbx) help stop" or "(dbx) help list" return the command to stop the debugger or a list of debugger commands for which you can get help. To see a summary of some common commands, try "(dbx) help most_used".

4. Running your program in the debugger: Use the command 'run'. For example: "(dbx) run" will run the executable you indicated when you have started the debugger using the command "dbx". If you need to supply command line arguments to the program, list them following the run command as you would do in a usual

run. For example: “(dbx) run file1 23 file2” will pass the arguments “file1”, 23, and ”file2” to the executable. Typing ’run’ will restart the execution from the beginning.

5. Setting breakpoints in the program: A breakpoint allows you to examine the state of the variables at a particular location in the program. The debugger will stop at this location every time it is encountered. To stop at a certain line number: give the command: “(dbx) stop at “linenumber”. For example: “(dbx) stop at 27” stops the process before the statement at line 27 in your source program is executed. Use the command “(dbx) stop function_name” to stop the process on entry to the function function_name. For example: “(dbx) stop in main” stops the process upon entering the main function of the program.

6. Listing the breakpoints that are set: The command “(dbx) status” will give you a listing of the breakpoints. A sample portion of the output would be:

   ```
   Process  3149: [4] stop at "/usr3/csa026/ScanAssign1/my.c":27
   Process  3149: [5] stop in main
   ```

7. Removing a breakpoint: The command “(dbx) delete number” removes the break point number, where number is found in the status list; the command “(dbx) delete all” removes all breakpoints. For example: (dbx) delete 4 removes the above breakpoint at line 27. In most implementations, it is sufficient to just use ’d’ rather than the ’delete’ so “(dbx) d 4” also removes the breakpoint at line 27.

8. Stepping through the program: You can examine every line as it is executed, including entering all functions. The command to use is ’step’ but ’s’ is usually sufficient. Giving the command “(dbx) s” when your process has been stopped by a breakpoint set at a line x will step ahead the execution to the next line x+1 of your code. If the next line is a function call, the debugger will go into the function and stop at the first line of that function.
   Example: assume that you run a.out that is stopped in main at line 20.

   ```
   Process  3510 (a.out) stopped at [main:20 ,0x10000f7c]
         20  Buf = &Buffer[0]; /* Buf = Buffer */
       (dbx) s
         Process  3510 (a.out) stopped at [main:22 ,0x10000f84]
         22  length = fill_buffer (Input, Buffer);
        (dbx) s
         Process  3510 (a.out) stopped at [fill_buffer:81 ,0x10001184]
         81  int i = 0, ch;
   ```

   Note: On function exit, it will return to the calling location.

9. How to examine code line by line, but not jumping into functions: Use the command is “next” but “n” is usually sufficient. So giving the command “(dbx) n” will step ahead to the next line of your code. If the next line is a function call, the debugger will execute the function and your control will go the the next line. Example:

```
    Process  3514 (a.out) stopped at [main:20 ,0x10000f7c]
    20  Buf = &Buffer[0]; /* Buf = Buffer */
  (dbx) n
    Process  3514 (a.out) stopped at [main:22 ,0x10000f84]
    22  length = fill_buffer (Input, Buffer);
  (dbx) n
    Process  3514 (a.out) stopped at [main:25 ,0x10000f98]
    25  while (!Done)
```

10. Jumping ahead to the next breakpoint: Use the command is “cont” or just “c” as in “(dbx) c”. If no further breakpoints are encountered you program will run to completion.

11. Examining the values of variables: Use the command “print” or just “p” to display the value of variables. For example: “(dbx) print Done” displays the current value of the variable “Done”; “(dbx) p Token” displays the current value of the variable ”Token”; “(dbx) p i,j,k” displays the current values of variables i, j, k.

12. Listing a portion of the source code: You may list a portion of code following the current line using the command “(dbx) list”; typing the “list” command repeatedly will scroll down the code. You may also the code starting at a certain line number. For example, “(dbx) list 10” source lines starting with line 10 are listed. As with “list” above, typing the ’list’ command repeatedly will scroll down the code. You may also list the code at the start of a function. For example, “(dbx) list main” will start the listing of the main function of the program. As before, typing the “list” command repeatedly will scroll down the code.

A more detailed manual of the debugger installed on your system is obtained by locating the debugger you have and then using the manual page of that debugger. For example, the debugger on the Linux machine is usually called gdb. Thus, a user manual for gdb can be obtained by typing “man gdb” on the system you are logged in.