Advanced FIELD Tutorial

Carolyn Duby
September 17, 1991

FIELD is a programming environment which integrates and provides graphical user interfaces for several tools in the UNIX environment. It is designed to be used by students in computer science classes as well as programmers participating in advanced research projects. The graphical user interface is easy to understand, and the system is configurable, making it flexible enough for both beginning and advanced users. This tutorial will give the advanced programmer an introduction to most of the tools supported by FIELD and quickly learn how to use the environment effectively to develop software. This paper is a step by step
guided tour through the writing an debugging of a simple linked list program.

Prerequisites

It would be helpful if the user of this FIELD tutorial had a basic knowledge of the X
window system and the UNIX environment including the following:

1. how to create windows
2. how to use the mouse and keyboard as input devices in X
3. how to use various graphical user-interface objects such as pull-down menus, dialog boxes, scroll bars, icons
4. how to write basic programs in C
5. how to manage programs with make

Preparing for the Tutorial

You will need to do the following things before you begin this tutorial:

1. Login to a Sun workstation.
2. Add the directory /cs/bin/field to the path variable in your .cshrc file. To do this edit the file called .cshrc with any editor. On the line that says `set path=(`, check to see if the directory /cs/bin/field appears between the open parenthesis. If it does not, add the directory /cs/bin/field just before the closing right parethesis. If you have modified the .cshrc, type `source .cshrc` on all shells. This will tell the system that you have made changes to this file.

3. Make a directory called field in your home directory by executing `mkdir field` from the shell.

4. Enter the new directory by typing `cd field`.

5. Copy the Files in the directory *tutorial_file_name* to the current dir by typing `cp` *tutorial_file_name* `/* .`

Getting into the FIELD environment

1. Start the X Window system by typing `xstart` or `xinit`.

2. Move the mouse to any window and type `field`. The Control Panel window like the one in Figure 1 will be placed on the screen.

The Control Panel Window and FIELD Help Features

The FIELD Control Panel gives you access to all the tools available in the environment. Each tool is represented by a lettered icon and can be opened by clicking on its icon. Online FIELD help as well as the UNIX shell can be accessed from the control panel.

1. FIELD windows can be moved, resized, iconized, pushed and popped. The following letters refer to the parts of the FIELD window in Figure 2.

   (a) Name of window
   (b) Pull-down menu bar
   (c) Press mouse down and drag here to move window.
   (d) Press mouse down here and drag to resize window.
   (e) Click mouse here to push window below other windows that it is covering.
   (f) Click mouse here to pop window to the top of other windows that are covering it.
   (g) Click here to iconize. Shift-click here to remove window.

2. Choose the Shell button from the UNIX menu. This should bring up a shell in your home directory from which you may execute UNIX commands. For instance, type the `pwd` command and see what the current directory is. See Figure 3.

3. Choose the Help button from the Commands menu of the Control Panel. Figure 4 shows the help window that should be displayed. The HELP Information tool of FIELD provides online documentation for the user. The two paneled windows beneath the title HELP Information contain a list of the topics which online help documentation is available for. The small window below the topics

displays the name of the help information that is being currently displayed. The scrolling window below that is the actual help information.

4. Click on control from the topics list to see the help for the Control Panel. The help information for the Control Panel is divided into topics. Each of these help topics represents a FIELD tool. For other help information windows, the topics describe the menu buttons or other features that are available.

5. Choose dbg from the topics list to see a description of what the debugging tool does.

6. Choose ^^control from the topics list to get back to the control panel help window.

7. Choose ^^field from the topics list to return to the field help window.

8. Choose Mouse Help from the Commands menu. This opens the help facility that provides information about the mouse buttons. Figure 5 shows the help window for the mouse buttons, which tells what the left, middle, and right mouse buttons are used for in a particular window or area of a window. For example, move the mouse into the help information window. A description of the actions of the mouse buttons should appear in the MOUSE HELP Information window.

Editing a File with the aedit Tool

The aedit tool is the editor that is used throughout FIELD.

1. Starting aedit from Control Panel

   To start aedit click on the aedit icon. A dialog should prompt you for the name of the file to edit. In the Name field, type *sorter.c* and click on accept. The current contents of the file are displayed. Since *sorter.c* is a new file, the window should be empty. Several features of the edit window are noted in Figure 7.

2. Adding text

   Adding text in the field editor is simple. Just type the letters from the keyboard. When you want to go to a new line simply press the return key. Notice that when you hit return after typing main(), the cursor is indented on the next line. This is the indenting feature of the field editor. It allows the user to

   format code quickly and easily. You can turn off the indentation feature by choosing the SetMode button from the Edit menu and deselecting the auto_indent radio button. The rest of this section describes how to perform a number of editing functions. Experiment with these functions as you use aedit to create the following C program:

```
#include "list.h"
#include <stdio.h>



main()
{
   list_item_t   *list_head;
   int            next_value;
   char           next_line[LINE_LEN];
   list_item_t   *curr_node;
   list_item_t   *new_node;
   list_item_t   *before_curr;

   list_head = create_new_item(NULL);
   printf("Please enter the next value to add to the list
           (enter ^d to quit):\n");
   while (gets(next_line) != NULL)
   {
      sscanf(next_line,"%d",&next_value);
      curr_node = FIRST_NODE(list_head);
      before_curr = list_head;
      while (!LAST_NODE(curr_node) &&
              (NODE_VALUE(curr_node) < next_value))
      {
         before_curr = curr_node;
         curr_node = NEXT_NODE(curr_node);
      }
      new_node = create_new_item(next_value);
      add_to_list(before_curr,curr_node,new_node);
      printf("Please enter the next value to add to the list
              (enter ^d to quit):\n");
   }
   free_list(list_head);
}
```

3. Moving throughout a file

   (a) Scrolling
   You can move forward and backward in a file by pressing the mouse on the up arrow of the scroll bar to move up or the down arrow to move down. To move greater distances, hold down and drag the white box on top of the grey area of the scroll bar between the arrows. Move the white box up to scroll backwards and down to move forwards.

(b) Moving the cursor

The cursor can be moved by clicking the mouse in the desired location or by choosing the GenericMove command from the Move menu. This will bring up a dialog box, like the one in Figure 8. It asks for the reference point of the move and the number of lines and characters to move. For example, to go to the beginning of the 5th line of the file, choose Start of File as the basis and set delta_lines to 5 and delta_chars to 0. The cursor can also be moved by the arrow cursor keys.

4. Searching for a string in a file

   Often while programming, it is helpful to be able to find all occurrences of a string in the program text. To do this choose GenericSearch from the Move menu. Type the string that you want to search for in the Pattern field of the resulting dialog box and then choose accept. The cursor will be moved to the first occurrence of the string after the current cursor position. Scrolling will occur if necessary. To insure that all occurrences are found, including the ones before the cursor, select the wrap-search radio button in the GenericSearch dialog box. This will cause the search to go to the beginning of the text after it has found the last occurrence of the string in the file.

5. Cutting Text

   A section of text can be removed from a file and held in a buffer. The buffer text can then be placed in a new position by pasting the text. Cutting text can be acheived by selecting GenericDelete from the Edit menu. The starting point and ending point can be chosen from a number of alternatives. For example, the user can select a some text by pressing down on the mouse and dragging. The selected text will be underlined. To delete the text from the file and place it in the buffer, set the from radio button to Start of Selection and the to radio button to End of Selection, and then click on the accept button.

6. Copying Text

   A section of text can be copied from a file into a buffer without removing the text from its original position. To do this choose GenericPick from the Edit menu. The dialog box is similar to the GenericDelete dialog box. To copy selected text into a buffer, set the from radio button to Start of Selection, set the to radio button to End of Selection, and then click on the accept button.

7. Pasting Text

   Text that has been Cut or Copied into a buffer can be inserted into the file by using the GenericInsert command from the Edit menu. There are a number of options for the placement of the text. For example, to insert the text in the buffer at the current cursor position, set the where radio button to Current Postion and then choose accept.

8. Inserting Blank Lines into Text

   When you hit the return key when the cursor is not at the end of the file, you will notice that blank lines are not added to the file. Putting the editor in Open mode will make it so that the return key adds a blank line to the text. To do this, press the control key and then the o character. Now the return key should cause blank lines to be placed in the file.

9. Saving a File A file can be saved to disk by using either the Savefile or Saveas command from the File menu.

   (a) SaveFile

       SaveFile saves the file to disk with the name that is shown in the upper left corner of the window to the right of the move symbol.

   (b) SaveAs

       Saveas writes the file to disk with a new name. Choosing SaveAs from the File menu brings up a dialog box asking for the new name of the file. Enter the desired name in the Name field of the SaveAs dialog box and choose the accept button.

10. Using the function keys

    Many of the function keys have special meaning in aedit. These meanings are documented in the Brown Workstation Environment manual.

11. Quitting the Editor

    (a) Closefile

        Choosing Closefile from the File menu causes the editor to automatically save the file and quit.

    (b) Closeabort

        Choosing CloseAbort from the File menu causes the editor to quit without saving the file. If the file has not been changed, the editor exits immediately. If the file has been modified, FIELD puts up a dialog box asking the user to confirm that the file should not be saved. If the Abort radio button is set and the accept button chosen, the file will not be saved. If the Save button is set, then the file will be saved before quitting aedit. The default button is Abort.

Building an Executable with the Build Tool

1. Click on the build icon. The window shown in Figure 9 should appear.

2. Setting the project

   The project represents the program currently being worked on. Choose Set Project from the Project menu. A dialog box like the one in Figure 10 should appear. Click on the accept button. Since you started field in the directory that your tutorial code is in your project should be in the current working directory.

3. Declaring File Dependencies

   (a) Setting the display
   Set the display to Information, by choosing the Information button from the Window menu. The new display should look like Figure 11.

   (b) Choosing the file to set dependancies for
   Choose SetFile from the File menu. A dialog box similar to the one pictured in Figure 12 should appear. Click on the file list.c and then choose accept.

   (c) Setting the Dependancies
   Click on the middle mouse button to the right of the word Dependencies. The dialog in Figure 13 should appear. The file list.c includes the header file list.h and therefore list.h is a dependant file, i.e. if list.h changes then list.c will have to be recompiled. Choose list.h from the list of files by clicking on it, and then choose accept. The file list.h should now appear under the word dependancies.

   (d) Setting Dependencies for sorter.c
   Declare that sorter.c is dependent on list.h by following steps 3(b) and 3(c). for the file sorter.c.

   (e) Creating the link command
   Set the file to sorter. Make it dependent on list.o and sorter.o. In the bottom window type the link command `$(CC) $(CFLAGS) $(LD_FLAGS) -o sorter list.o sorter.o` telling the make command to link the files list.o and sorter.o together to make an executable program.

   (f) Setting the default program to build
   Choose Options from the Project menu. Type the name sorter in the Default Build field. This will allow you to make sorter by simply choosing Make Default from the Command menu.

   (g) Saving the File Relationships
   Choose Save from the Project menu to store the dependencies that you have just set. The dependencies are kept in a database so that the next time you start field, you will not have to redeclare them.

4. Graphical Views of File Dependencies

   File dependencies can be represented as a directed graph with each node representing a graph and each arc representing that the source node is dependent on the destination node. To view the dependency graph for a file, first choose Graph from the Windows menu and then set the file using the Set File button from the File menu. The dependency graph for sorter is shown in Figure 14.

5. Compiling Files

   (a) Open Sorter.c with the annotedit Tool

       The annotedit tool is an editor like aedit, but it also has the capability of marking the lines of a file with graphical symbols for compiling and debugging. To open sorter.c, click on the annotedit icon on the control panel. A dialog box should appear asking for the name to the file to edit. Type *sorter.c* in the Name field of the dialog box and click on the accept button. A window like the one shown in Figure 15 should appear.

   (b) Set the Build Display Window to Transcript

       To see what commands are being executed by build choose Transcript from the Windows menu of the Build tool.

   (c) Compiling a File

       Choose compile from the Commands menu. A thin message bar should appear saying that it is compiling your file. After the compilation is done, a dialog should display any compilation errors. Click the accept button on the dialog box when you are finished reading the error messages. If there were any compilation errors or warnings, an E or W enclosed in a box will appear to the left of the line containing the error.

   (d) Displaying Compilation Errors

       To display the error message for a line, click the middle mouse button on the E or W annotation. A dialog box similar to Figure 16 should appear telling what the error message is. After you have finished reading the error message, click on the accept button of the error dialog box.

   (e) Correcting Compilation Errors

       To correct the compilation errors, you can use the any of the editor commands described in the editing section.

   (f) Removing an Error or Warning Annotation

       After you have corrected a compilation error or warning, remove the annotation to the left of the line by clicking the middle mouse button on the annotation, clicking on the box to the left of the word remove, and then choosing accept. This is useful for keeping track of which compilation errors have been fixed and which have not.

   (g) Producing a Valid .o File

       Repeat the above steps until sorter.c and list.c compile correctly.

6. Making the Final Running Version

   To link all the of the files together, choose MakeDefault from the Command menu of the Build tool or from the Commands menu of annotedit. A message similar to the one shown in Figure 17 should appear in the build transcript window.

Debugging with the dbg Tool

The field debugger is very similar to dbx on the Suns. It has a window where commands can be entered with the keyboard, buttons that can be clicked on for controlling the debugger, and graphical breakpoint annotations that can be set in annotedit.

1. Click on the dbg icon on the control panel. A dialog box and window should appear asking for the name of the program to debug. Type *sorter* in the Debug field and click on the accept button. The debugger should start up and look like Figure 18.

2. Setting Breakpoints

   A breakpoint represents a line in the source file where the debugger will stop executing. It can be used to stop the program periodically so that the status of variables can be checked and the problem area isolated.

   (a) Open annotedit sessions on sorter.c and list.c, if you do not already have them open.

   (b) Check the annotation type selector to the right of the program text to make sure that it is set to BREAK.

   (c) Go to line 20 of sorter.c and press the right mouse button down in the annotation column to the left of the source file. A stop sign annotation should be placed in the annotation column. See Figure 19.

3. Executing Programs in the Debugger.

   The debugger has several different program execution modes. It can step through the program one line at a time or run the program continuously until it hits a breakpoint or the program exits.

   (a) Starting Execution from Beginning of Program

   To begin running the program continuously from the begining of the program click on the run button at the bottom of the debugger window. The program should now ask you to enter the first value in the list. Type an integer followed by a carriage return in the debugger window below the prompt. After entering the value the program should run until it hits the breakpoint that was set at line 20 of sorter.c. There should be an arrow annotation at this line and a message in the debugger window saying that it stopped at the breakpoint. See Figure 20.

   (b) Starting Execution after a Breakpoint

   To begin running the program again after a breakpoint press the Continue debugger button. The program should now continue until it comes to line 20 again.

(c) Executing Programs one Instruction at a Time

After Run has been pressed and the program has stopped at the breakpoint at line 20, the debugger can now begin execution one line at a time. Pressing the Step debugger button will cause the debugger to execute the next line of the program and stop. The arrow showing the position of execution will move down one line and the debugger will display a message noting the next line to be executed. Keep pressing step until the debugger reaches line 28 of sorter.c which involves a function call. If step is pressed then the program execution will continue at the first line of the function create_new_item. To avoid jumping into that function, press the Next button. The call to create_new_item will be executed and the debugger will go onto the next line of sorter.c.

(d) Stopping Execution

If the debugger is executing a program and you would like to halt it but there is no breakpoint set, press the Stop debugger button. The debugger will display a message telling what line it stopped on. This feature is useful for breaking out of infinite loops.

4. Printing the Values of Variables

The debugger allows you to see the value of variables. While the debugger is stopped select the name of a variable in the current procedure and press the Print It debugger button. This will print the value of the variable in the debugger window.

5. Tracing

Placing a trace at a line in the source will cause the debugger to print out a message each time that line is reached. In annotedit, set the annotation type to trace by clicking on the TRACE button to the right of the source code. To place a trace at a line 29 of the program, press the left mouse button in the annotation column to the left of the source code at line 29 of sorter.c. Now press the Run debugger button. Each time the debugger executes line 29 it will print a message saying that it has reached that line.

6. Removing Breakpoints and Trace Annotations

Debugging annotations can be removed by setting the annotation type to the type of annotation that you want to remove and then clicking the left mouse button on the annotation, or clicking the middle mouse over the annotation, clicking on the remove box of the dialog box that comes up, and then clicking on the accept button.

7. Other Commands

Other commands are available for the debugger. These commands can be entered at the ddt prompt. Read the dbx manual page for more information about the complete set of commands.

Using Graphical Views of Programs

1. Displaying Program Structure with flow

   The flow tool will automatically produce a directed graph representing the procedure calls in the program. A node in the graph represents a function and an arc represents that the source function calls the destination function. While you have a debugger session open for sorter, click on the flow icon. A flow graph for the sorter program should appear and look like Figure 21.

2. Displaying Program Stack with viewstack

   The program stack represents the function calls that a program has made. As the program runs the stack display will automatically update itself.

   (a) Open a debugger session on the file sorter.
   (b) Open an annotedit on sorter.c if you have not already.
   (c) Click on the viewstack icon from the Control Panel. A window similar to the one shown in Figure 22 should appear.
   (d) Set a breakpoint at the first line of the function main.
   (e) Choose run from the debugger buttons to execute until the first breakpoint.
   (f) The stack viewer should have been updated and the new view should look like Figure 23. Each function call entry on the stack is offset by a number representing the stack level. The highest number appears at the top of the screen and represents the current procedure that is executing. Each stack entry also includes the parameters to the function in parenthesis and the line number and source file that the function is defined in followed by a list of the values of any local variables.
   (g) Choose step from the debugger buttons and an entry in the stack should be produced for the function call to create_new_item. The stack will continue to be automatically updated as the program runs.

3. Tracing Values of Variables with viewtrace

   The viewtrace tool allows the user to monitor the values of a variable. Each time the variable is changed, the viewtrace display is updated.

   (a) Open a debugger session on sorter.
   (b) Click on the viewtrace icon on the control panel.
   (c) Choose the Options button from the View menu. A dialog box should ask for the name of the variable to trace. Enter next_value in the Variable field of the dialog box and click on the accept button.

(d) Choose Run from the debugger buttons. Each time you enter a new value to be added to the list, the next_value trace display should be updated. Figures 25 and 26 show the viewtrace and debug windows after Run has been pressed but before the first value is entered, while Figures 27 and 28 show them after the value 12 has been entered.

4. Displaying Data Structures with typeedit and display

   FIELD can graphically display the values of variables. This is especially helpful for viewing high level data structures, such as linked lists and trees.

   (a) Displaying a Variable with the display Tool

   Open a debugging session on sorter and add a few values to the list. Press the Stop debugger button while the program is waiting for input to discontinue execution. Choose display from the control panel. A dialog box will appear asking for the name of the variable to display. Enter list_head in the Variable field. A picture similar to the one in Figure 29. This is called the default display.

   (b) Changing How a Variable is Displayed with the typeedit Tool

   i. Click on the typeedit icon. A dialog box should prompt for the type to be edited. In the Type field, enter struct list_item_t * and choose the accept button. A window similar to the one in Figure 30 should appear.
   ii. Choose Default from the Top menu to see how the default view is displayed.
   iii. Choose Clear from the file menu to erase the default display and start a customized one.
   iv. Make the new display a tiled object, by choosing Tiled from the Top menu. This tells the editor that the C type can be represented as a rectangle divided into several different subregions called tiles.
   v. Press down on the mouse in the upper left corner of the large rectangular white region and drag it to make a tile the width of that region. Make another tile like it directly below. The resulting view should look like Figure 31.
   vi. Click the middle mouse button in the upper tile to highlight it. Choose box from the Edit menu and then click on the word value in the column to the right of the display.
   vii. Click the middle mouse button in the lower tile to highlight it. Choose Pointer from the Edit menu and then click on the word next in the column to the right of the display. The window should now look like Figure 32.
   viii. Save the new display for the linked list by choosing Save from the File menu. To make the display window reflect the new display method,

choose Update from the Display menu of the list_head display. The resulting linked list should look something like Figure 33.

(c) For efficiency reasons, the display graphics do not update themselves automatically, but you can add update annotations at lines of code where the variable changes. To do this, open annotedit session on sorter.c. Set the annnotation type to UPDATE. Click the left mouse button in the annotation column to the left of line 30 of sorter.c. Choose continue from the debugger buttons and enter values when the program prompts you. The graphics display should update itself automatically after the new value is added to the list.

Figure 1: The FIELD Control Panel

Figure 2: FIELD Windows

Figure 3: FIELD UNIX Shell

Figure 4: FIELD Help Information Window

Figure 5: Mouse Help

Figure 6: HELP information for the Control Panel

Figure 7: The aedit Tool

Figure 8: Dialog Box for GenericMove

Figure 9: The build Tool

Figure 10: Dialog box for setting Project

Figure 11: Information display of the build tool

Figure 12: Dialog box for setting File

Figure 13: Dialog box for setting File Dependencies

Figure 14: Dependency Graph for sorter

Figure 15: The annotedit Tool

Figure 16: Dialog Box Showing Compilation Error

Figure 17: Transcript of Load Message for Build of Sorter

Figure 18: The dgb Tool

Figure 19: Annotedit with Breakpoint set at line 20

Figure 20: Execution of Sorter

Figure 21: Call Graph for sorter

Figure 22: Viewstack Tool before Execution

Figure 23: Viewstack after Call to create_new_item

Figure 24: Options Dialog Box for viewtrace

Figure 25: Debugger View Before Entering 12

Figure 26: viewtrace Before Entering 12

Figure 27: Debugger View After Entering 12

Figure 28: viewtrace After Entering 12

Figure 29: Default Display for list_head

Figure 30: typeedit Representation of Default Display for list_head

Figure 31: Tiled Region after Clearing and Making New Tiles

Figure 32: typeedit after Completing Edit of Display

Figure 33: Display of list_head after Update