INTRODUCTION

Computing technology is evolving rapidly, and remarkable improvements in hardware capabilities are changing the way that people use computing. Portability has always been a concern and continues to be a major factor in determining the success of an application. Many users will soon have desktop systems that provide power that was found only in mainframe systems just a few years ago. The interconnectivity of machines is improving, and most installations now consider heterogeneous local area networks to be a standard part of the computing infrastructure.

Operating systems are changing to adapt to this multivendor environment, emphasizing the need for similar changes in application software. Applications developed using the SAS® System on the UNIX operating system and derivatives are easily portable because they inherit the portability found in both the UNIX operating system and SAS software.

This paper highlights some of the specifics that make the UNIX operating system and SAS software a powerful combination for writing portable applications. A sample application is presented and then modified to illustrate how to develop an application that is portable and yet able to interact closely with the host operating system.

UNIX MAKES IT EASY TO WRITE PORTABLE APPLICATIONS

The popularity of the UNIX operating system has grown tremendously in the past few years and continues to rise. Several things make the UNIX environment the right choice for all kinds of computing.

The philosophy that drove the design of UNIX has provided us with systems that are powerful and flexible. The UNIX operating system has been adapted to run on a large variety of hardware. For smaller machines, Apple provides A/UX for the Macintosh, Commodore® provides SysVR4 for the Amiga, and there are several versions based on the Intel 80386 CPU. The UNIX operating system has been the preferred choice on mid-sized workstations for years. IBM® has developed AIX®, which runs on PCs, workstations, and mainframes.

This choice of hardware platforms allows users to select a UNIX operating system that matches their needs in terms of cost and capability. For example, several small desktop machines can work with one or two larger systems to provide a department with the tools it needs for communications, data processing, and other tasks.

Most applications written for one UNIX platform can be transported to another UNIX platform with little or no change. This is true for several reasons. One of the design goals of the UNIX operating system was to remove dependencies on particular devices. Applications are written to work with a device driver, and the driver hides the various details of using a particular device. Also, the C language, which is very portable, is the standard programming language on the UNIX operating system. In fact, much of the UNIX operating system is written in the C language. C compiler products are available for most operating systems because of the popularity of the C language.

The UNIX environment provides a number of con-
structs that allow flexibility in the way that an application organizes and processes the data it uses. The input and output parts of a program are written with system calls that use a standard logical device. When the program is run, this standard device can be a disk or tape device or a pipe. Pipes allow one program to pass data directly to another. With the standardized system calls that the UNIX operating system provides, the application writer does not have to be concerned with the details of the reading and writing data. The UNIX environment also offers the ability to run programs easily in a background mode, or on some remote UNIX system.

**SAS SOFTWARE MAKES IT EASIER TO WRITE PORTABLE APPLICATIONS**

The SAS System provides application development software that is portable to many versions of the UNIX operating system, and to other operating systems on microcomputers, minicomputers, and mainframes. The various products that comprise SAS software allow an application developer to integrate a wide range of capabilities. Some examples include data entry with customized forms and validation of input, analysis, and presentation for statistical analysis. Other areas and capabilities are covered by SAS/INSIGHT®, SAS/OR®, SAS/PUBLISH®, SAS/ACCESS® Software and the other software from SAS Institute.

**GRAPHICAL USER INTERFACES MAKE IT EASIER STILL**

Release 6.07 of the SAS System for the UNIX Environment includes graphical user interfaces based on the X Window System from the Massachusetts Institute of Technology. Interfaces are available that support both the OSF/Motif and OPEN LOOK styles. These interfaces affect the way that SAS Software is used for the following reasons.

- **High-resolution graphics can display more information.**

  The bitmapped graphics displays used with the X Window System allow more information to be displayed than is possible on character cell devices. For example, the icons used with SAS/ASSIST® software are much more distinctive than the text-based buttons. This difference makes the product easier to use.

  Also, the user can display high-resolution graphics in a separate window on the same device used for the SAS Display Manager System. This makes graphics displays easier to create and modify. Other, nonwindowing devices often have limitations that prevent display manager and graphics windows from being displayed simultaneously.

- **The X Window System interfaces offer a point-and-click style of interaction.**

  The point-and-click capability makes interacting with the display manager much easier. This is immediately evident when using SAS/ASSIST software. A few button presses can produce custom reports and analyses with a fraction of the effort that it would take to write the code by hand.

- **X is a network-based windowing system.**

  This means that compute and display processing can occur on two different machines on a network. Several relatively inexpensive display devices can access applications that reside on a more powerful compute server, and the only
necessary connection is some type of network that can support one of the X transport mechanisms.

The compute system and the display device can be separated by ten feet, ten miles, or a thousand miles. In addition, the X protocol will work across different operating systems. For example, a user with a VAXstation can interact with an application on a UNIX operating system just as easily as a user with a UNIX workstation.

- Clients are easy to customize.

Programs written for the X Window System (usually referred to as clients) can be customized through the use of X resources. These resources allow the user to customize both appearance and behavior. For example, the colors and font used by the display manager can be changed, allowing each user to customize a SAS session to suit his or her preferences.

One of the SAS application resources indicates which actions should be associated with the various keys. For example, the Ctrl-E key could be set up so that it deletes all of the text on a line from the cursor location to the end of the line.

More details on using resources with the X interface to the SAS System can be found in the SAS Companion for the UNIX Environment and Derivatives, Version 6, First Edition.

- Interclient communication is enhanced.

X clients can exchange text selections with each other, providing a useful cut-and-paste capability; the X interface to the SAS System supports this capability. Indicate a text selection with a click-and-drag action. Then paste the selection into another X client, or another SAS session with a single mouse click. Text can also be copied from most X clients (including xterm into the SAS editor.

SCREEN CONTROL LANGUAGE (SCL) AND THE FRAME ENTRY

SCL provides the ability to develop powerful applications. Because the SAS System is portable, applications can be developed and tested in a low-cost environment and then moved into production.

SCL is probably best described in the introduction of SAS Technical Report P-199, Using SAS Screen Control Language in Release 6.06.

Screen Control Language (SCL) is a programming language that you can use with SAS/AF® and SAS/FSP® software to assist your development of interactive applications. You can use SCL to

- manage an application’s windows
- perform cross-field validation of values
- manipulate data values
- manipulate SAS data sets
- manipulate external files
- control the application environment
- submit statements to the SAS System for execution.

The details of using SCL and the FRAME entry are not covered in this paper. Documentation on SCL can be found in SAS Screen Control Language: Reference, Version 6, First Edition and SAS Technical Report P-199, Using SAS Screen Control Language in Release 6.06. The FRAME entry is documented in SAS Technical Report P-226, SAS/AF Software: Creating
The FRAME entry capability added to SAS/AF software in Release 6.07 makes application development much easier. The interactive nature of FRAME entries facilitates the iterative nature of user interface design.

**A SAMPLE APPLICATION**

An example of an application that uses SCL and the X interface to Release 6.07 follows. Let's start with an existing application and then discuss the steps needed to modify it to add additional functionality.

The application helps a system administrator keep track of resource usage across multiple systems from different vendors. The main screen is shown in Figure 1.

![Figure 1: The Main Screen in the Sample Application](image)

There is a large button for each of three operating systems (UNIX, VMS, and MVS); a smaller button is used to exit the application.

Select one of the large buttons to bring up the Configuration Screen for that operating system.

**Figure 2: The UNIX Configuration Screen**

Clicking on the UNIX button causes the UNIX Configuration Screen to appear. This screen, shown in Figure 2, has three main areas. Across the top are check boxes for different types of measurements. Only one of these may be selected at a time. There are two list boxes on the left side of the screen. One is marked Available and contains a list of all of the UNIX systems on the network. Click on a system name in this list. The name is copied over to the Selected list. Click on a name in the Selected list to remove it from that list.

Use the buttons in the third area on the right to go to either the Data screen or to the Main screen.

The selections for the system names and the measurements are managed by the SCL entry for this screen. The selected system name and data type are stored as SCL variables and are passed on as needed.

The Data Display Screen contains a graphical display of the measured data. This screen is
discussed in more detail below.

IMPROVING THE SAMPLE APPLICATION

This application as described is useful for some tasks, but it could be improved in a number of ways. This section focuses on one change: adding an additional data type.

The application currently provides information on CPU and network usage. Suppose you would like to add an option to examine how much disk space is being used by users' home directories.

No change is needed in the main screen. The UNIX Configuration Screen needs a way to specify disk space. For this, add another checkbox in the Measurements area, using the Copy option in the popup menu in the FRAME entry.

The FRAME entry type uses popup menus to provide access to the many operations that it provides. Move the pointer to the NetworkTraffic checkbox, and press the right mouse button. When the Actions popup menu appears, select Copy. An outline of the region appears. As the pointer is moved to the area to the right of the checkboxes, the outline moves with it. Press any mouse button to place the new region.

Next, press the left mouse button to make the region the active region, and the region is highlighted.

Press the right mouse button to bring up the Actions menu again, and select Object Attributes to bring up the Checkbox Attributes dialog window. Figure 3 shows this screen with the dialog window.

When the dialog window appears, click the left mouse button in the Name field, and change the name of this check box to DSKBTN. Click the left mouse button in the Text field, enter the text label Disk Space. Select the OK button to exit the dialog window. Figure 4 shows the resulting screen.

Some additional SCL source is needed to deactivate the other checkboxes when this one is selected. The name given to the check box object is DSKBTN. When the check box is select-
ed, the code at the DSKBTN label is executed. In this case, we want to call the ACTIVATE methods for the check boxes named CPUBTN and NETBTN. The SCL code to deactivate the other two check boxes follows:

```scl
DSKBTN:
call notify("CPUBTN", '__ACTIVATE__', 0);
call notify("NETBTN", '__ACTIVATE__', 0);
return;
```

The last screen to be considered is the Data Display Screen. Since this screen hasn't been presented yet, a brief digression is in order.

Figure 5 shows the UNIX Data Display Screen. The System Name and Data Type to be measured are shown in the top part of the screen. The main area is used for a graph, showing the results of the measurements.

![Data Display Screen](image)

Figure 5: The UNIX Data Display Screen

At the bottom are various buttons that are used to update the measurements and the associated graph or to move to one of the other screens.

The Update button accesses the selected host system to get the requested data; the graph is be updated automatically. The Report button brings up another screen where the data obtained during the update can be examined in more detail. Use the Config button to return to the Configuration Screen.

Now we can resume discussing the changes we need for the added function in the Configuration Screen.

The screen itself needs no change, but the SCL code associated with the entry needs to be modified to handle the new data type.

This code will use UNIX remote shells and the du command to get the data on disk usage from the selected system. The data are read into a SAS data set, and the graph is updated to reflect the new data.

The SCL needed to obtain the disk space usage data, and to update the graph is listed in Appendix 1. The output from the du command is read through a filename which has been assigned to a pipe. Once the data has been read into a SAS data set, the GCHART procedure is used to display the new graph.

**Using the Modified Application**

Now let us go completely through the revised application, to see how it is used to examine disk space on a UNIX operating system. From the Main screen, select UNIX, then Configure. On the UNIX Configuration Screen, select a system called aobar and select User Disk Space as the item to measure. Press the Return button to go back to the Main screen. Select Data to go to the Data Display screen.
When this comes up, select Update to get the data on disk space usage on a/obar.

When the new data has been read, the bar graph is updated. Look at the specific numbers by selecting the Report item.

Use the Config button to go back to the Configuration screen.

CONCLUSION

Portability across heterogeneous environments is often a critical consideration when developing applications software. However, this is easier to achieve than it has been in the past. Open systems, such as the UNIX operating system, promote greater consistency across different platforms. Applications development software, such as SAS/AF software, and SCL reduce the time needed to write and maintain applications.

The design of the visual interface itself has been very expensive. In Release 6.07, the FRAME entry allows the interactive design of user interfaces, significantly reducing development costs.

Trademarks

SAS, SAS/ACCESS SAS/AF, SAS/FSP, SAS/INSIGHT, SAS/OR and SAS/PUBLISH are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.
Appendix 1: SCL code excerpt to run UNIX commands

/* Construct the UNIX command string */
cmdstr = 'rsh ' || sysname || ' du -s /users/*';

/* Assign the filename to the pipe */
rc = filename('usage2', cmdstr, 'pipe');
if (0 = rc) then _msg_ = sysmsg();

/* Run a data step to read the output from du */
submit continue;
data unxdat.&mtype;	infile usage missover expandtabs;
length path $30. uname $8.;
input arnt path $;
label arnt = "blocks used" path = "Pathname"
  uname = "User";
  uname = scan(path,1); /* 2nd arg depends on the path name */
run;
endsubmit;

/* build name of catalog entry */
gcname = 'unxdat.' || &mtype || '.' || sysname || '.gseg';

/* Check for a current entry for this graph. If there is one, then delete it */
if (exist(gcname)) then do;
  rc = delete(gcname, 'catalog');
  if (0 = rc) then _msg_ = sysmsg();
end;
submit continue;
/* Display a horizontal bar chart graph of the data */
goptions reset=(axis, legend, pattern, symbol, title, footnote)
  norotate hpos=0 vpos=0 htext= ftext= target= gaccess=
  gsfmode= device=xcolor ctext=yellow graphrc
  interpol=join ftext=simplex;
/* turn off display so that we don't create a separate GRAPH window */
goptions nodisplay;

pattern1 value=SOLID;
axis1 color=yellow width=2.0;
axis2 color=yellow width=2.0;
axis3 color=yellow width=2.0;
proc gchart data=unxdat.&mtype gout=unxdat.&rntype;
  hbar uname / name='&sysname' maxis=axis1
    raxis=axis2 patternid=midpoint
      type=SUM sumvar=AMT;
run; quit;
goptions display;
endsubmit;

/* clear old and display new graph */
call notify('GRAPH1', '_CLEAR_');
call notify('GRAPH1', '_SET_GRAPH_', gcname);
return;

Using SCL to Build Portable UNIX Applications 1505