1. ABSTRACT

This panel discussion will focus on the use of the SAS/C™ compiler in the IBM mainframe environment. The panelists are all experienced software developers and have used SAS/C in a wide range of applications. The topics to be discussed span the spectrum of software development, from high-level system architectural considerations to interfacing with system-level services provided by the operating system.

2. PANELISTS

Ron Higgin is a Senior Scientist with Boole & Babbage, Inc. He is responsible for the overall architecture, design, and development of certain Boole & Babbage products. He has been an active participant in the GUIDE user group where he has held task force, project, and group management positions. Ron has also published papers on a wide range of topics pertinent to large users of IBM computing systems.

Carl Kass is a Senior Software Programmer with SyncSort, Inc. His current responsibilities include development of a parser generator, parser, code generator, and optimizer. He has a Masters degree in Computer Science from Columbia University and has also taught data structures, algorithms, and assembler classes at Columbia.

Larry Schumacher is a Senior Engineer with Stagg Systems, Inc. His current responsibilities include operating systems interfaces and integrating SAS/C code with an existing software base. Larry has a B.S. degree in Electrical Engineering Technology from Southern Illinois University.

Keith Watts is a Software Engineer with Boole & Babbage, Inc. He began his career as a software developer for IBM in Poughkeepsie and Santa Teresa. He has also worked for Intel, Crocker Bank, Bank of America, and Candle Corporation. Keith is now involved in a large C programming effort at Boole & Babbage.

3. PANEL DISCUSSION TOPICS

Ron Higgin:

"C" - An MVS Systems Language?

1. Background

Performance software design and development
IBM and PCM marketplace
MVS, VM, and VSE
Major IMS and CICS products
Most products written in S/370 assembler
Assembler inappropriate for many functions
Difficult to maintain and debug

2. Objective

Develop products in a high-level language

3. Requirements

Able to assume most assembler duties
Clean interface to other languages
Able to execute in any addressing mode
Flexible module packaging
Easy to distribute compiled code
Portable

4. Problems

Function packaging
Non-standard MVS execution modes
Access to MVS system services
Macro language

5. Implementation

Elements
- Externalized "C" run-time environment
- Built-in MVS service interfaces
- Function packages
- Run-time requirements
- Stable interface to compiled code
- No special compile options
- Replaceable
Run-time requirements (continued)
Symbolic debugger support
Multi-tasking and coprocess support

Run-time Environment
External to load module
Pre-allocated/dynamic create
External variable support
Buildin MVS service interfaces
Storage management
Program management
Other services

Function Packages
Definition
Program manager support
Function invocation
Function pack generation

6. Conclusions

"C" is an MVS systems language
Not perfect
Requires
A good macro language
Formal debugging "hooks"
SYM records in OBJ

Carl Kass:

1. Runtime Efficiency of SAS/C

Carl discussed the size of object code (and hence the execution speed) of the code generated by Release 3.00F and 3.01H of the SAS/C compilers, as compared to each other and hand-coded assembler. He also presented examples of object code that the compilers did a good job of producing and some in which a poor job was done. The use of register variables was also discussed.

2. Why did SyncSort, Inc. Choose SAS/C?

The merits of coding in a high-level language versus coding in assembler was discussed. The reasons for SyncSort's choice of C, as opposed to any other high-level language, were also presented. And finally, SyncSort's reasons for choosing SAS/C were presented.

Larry Schumacher:

1. Use of the SAS/C compiler 'INDep' option to interface with an existing multi-tasking software base.

The 'INDep' (independent library) option available with the SAS/C compiler allows a software vendor to integrate C programs into an existing software base. This option allows for a user routine (L&UPREP) to be called upon entry to a C function to locate or possibly allocate the C 'main' environment if the calling program is not a C function.

A large scale multi-tasking application may incur considerable overhead from allocation of a C 'main' environment on demand. By preallocating a permanent subtask and C 'main' environment, this overhead can be reduced. L&UPREP need only find the C 'main' environment for the currently executing task.

2. Use of the SAS/C source level debugger in a multi-tasking environment.

The SAS/C source level debugger can be activated/deactivated on a subtask level by passing the appropriate run-time options from the parent task when activating the subtask. The C 'main' entry point for the subtask receives the run-time options and then activates/deactivates the debugger without affecting other subtasks.

Keith Watts:

C portability was analyzed with a very special emphasis on differences between C subroutine libraries. Also included was an item-by-item description of major portability problems.

4. PANELIST CONTACT

Ronald W. Higgin
Boole & Babbage, Inc.
510 Oakmead Parkway
Sunnyvale, CA 94086

Carl Kass
SyncSort, Inc.
60 Tice Blvd., CN18
Woodcliff Lake, NJ 07675
(201) 930-9700

Larry Schumacher
Stagg Systems, Inc.
901 Threadsewde, Suite 150
Houston, TX 77079

Keith Watts
Boole & Babbage, Inc.
510 Oakmead Parkway
Sunnyvale, CA 94086

SAS/C™ is a trademark of SAS Institute Inc.