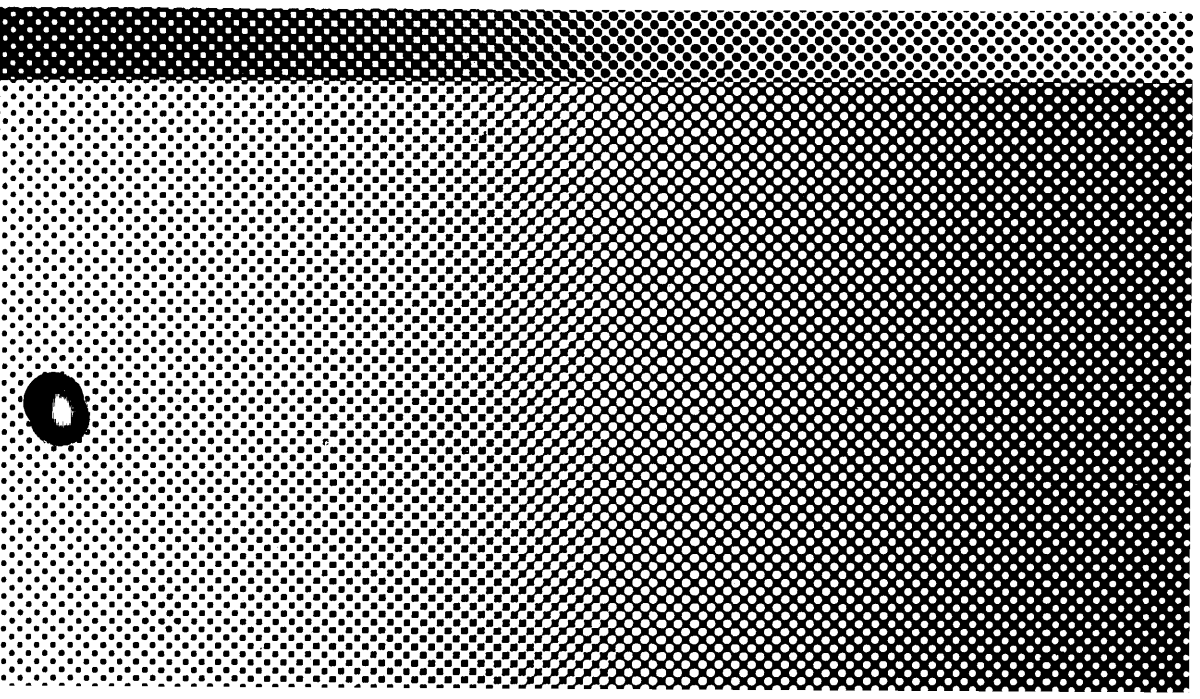
307-184

AT&T 3B2 Computer
UNIX™ System V

Advanced Programming Utilities
Issue 1

Release Notes



NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

Table of Contents

Introduction 1
Overview 1
Conventions Used in This Document 2

Contents of the Release 3

Software Features 5
Advanced C Utilities 5
Shared Library Upward Compatibility 6
Extended Software Generation Utilities 7
Source Code Control Utilities 8
SCCS Commands 8

Software Installation Information 10
Prerequisites 10
Software Dependencies 10
Storage Requirements 10
Installing APU 11

Software Notes 12

Documentation 16
Related Documents 16
How to Order Documents 17

List of Figures

Figure 1: Advanced C Utilities 3

Figure 2: Extended Software Generation Utilities 4

Figure 3: Source Code Control Utilities 4

Introduction

Overview

These *Release Notes* contain information about the Advanced Programming Utilities (APU). APU is a set of tools that are useful to programmers who

- do extensive programming in the C language,
- need tools to do advanced programming and symbolic debugging,
- want to create shared libraries,
- or work in an environment where it is necessary to track and maintain versions of files and programs.

APU includes the following packages:

- Advanced C Utilities, containing tools such as **cxref**, **ctrace**, **cflow**, and **lint** for the C language programmer, plus libraries, and **mkshlib** (to create shared libraries)
- Extended Software Generation Utilities, containing tools such as **m4** (a macro processor), **yacc** (Yet Another Compiler-Compiler), **sdb** (a symbolic debugger), **lex** (a generator of lexical analyzers), and **make** (a program construction tool)
- Source Code Control Utilities (SCCS), a system used to track changes made to files and to maintain a record of all versions

APU runs on any model of the AT&T 3B2 Computer running UNIX System V Release 2.0 or later releases.

These *Release Notes* contain the installation procedure for APU, a description of available documentation, technical information, and a description of new features.

Conventions Used in This Document

In this document, certain typesetting conventions are followed when command names, command line format, files, and directory names are described. There are also conventions for displays of terminal input and output.

- You must type words that are in **bold** font exactly as they appear.
- *Italic* words are variables; you substitute the appropriate values. These values may be file names or they may be data values.
- CRT or terminal output and examples of source code are presented in `constant-width` font.
- In output and source code examples, a backslash (\) at the end of a line indicates that the line wraps around without a break.
- A command name followed by a number, for example, **prof**(1), refers you to that command's manual page, where the number refers to the section of the manual. These manual pages appear in the *Programmer's Reference Manual*, unless otherwise noted.

Contents of the Release

APU comes on three diskettes:

- Advanced C Utilities, Issue 4, on 1 diskette
- Extended Software Generation Utilities, Issue 4, on 1 diskette
- Source Code Control Utilities, on 1 diskette

The directory structure and files are presented in the following tables.

DIRECTORY	FILES			
/bin	**mkshlib**			
/usr/bin	**cb** **cflow**	**ctc** **ctcr**	**ctrace** **cxref**	**lint** **regcmp**
/usr/lib	**dag** **flip** **lint1** **lint2**	**llib-lc** **llib-lc.ln** **llib-lm** **llib-lm.ln**	**llib-port** **llib-port.ln** **lpfx** **nmf**	**xcpp** **xpass**
/usr/lib/ctrace	**runtime.c**			
/usr/options	**acu.name**			

Figure 1: Advanced C Utilities

DIRECTORY	FILES		
/usr/bin	lex mcs	prof sdb	yacc
/usr/lib	libg.a libl.a	liby.a sdbp	sdbs yaccpar
/usr/lib/lex	ncform	nrform	
/usr/options	esg.name		
/bin	make		
/etc	install		

Figure 2: Extended Software Generation Utilities

DIRECTORY	FILES			
/usr/bin	admin cdc comb delta	get prs rmdel sact	sccsdiff unget val vc	what
/usr/lib/help	ad bd cb cm	cmds co de default	ge he prs rc	un ut vc lib/help lib/help2
/usr/options	sccs.name			

Figure 3: Source Code Control Utilities

Software Features

The following paragraphs contain brief descriptions of the features and some of the commands in this issue of APU. You may be familiar with some of these features, while others may be new to you. Even though this is the first release in which all of these features are part of the same product, most of these features have been available in other software packages.

Advanced C Utilities

Below are some of the Advanced C Utilities and their functions:

cxref is a C cross-reference listing generator

ctrace is a statement-by-statement execution trace facility

cflow produces a graph of program dependencies

lint detects faulty and non-portable code.

cb displays the structure of code

regcmp compiles regular expressions

All of these tools are described in the *UNIX System V Programmer's Guide* and the *UNIX System V Programmer's Reference Manual.*

The Advanced C Utilities package also contains **mkshlib**(1) (make shared library), which is used to create a shared library. Shared libraries are a feature of UNIX System V Release 3.0 that allow several **a.out** files to simultaneously use the same object code. The **mkshlib** command has options that allow you to specify the shared library specification file (which contains all the information necessary to build the shared library) and to name the host and target shared libraries. **mkshlib** and the shared library feature are described in detail in the "Shared Libraries" chapter of the *UNIX System V Programmer's Guide.*

Shared Library Upward Compatibility

Shared library compatibility is an important issue. These paragraphs explain how to build upward-compatible shared libraries. For more detailed information, see the "Shared Libraries" chapter in the *UNIX System V Programmer's Guide*.

Comparing Previous Versions of the Library

Shared library developers normally want newer versions of a library to be compatible with previous ones. **a.out** files will not execute properly otherwise. There are procedures that let you check libraries for compatibility. In these tests, two libraries are said to be compatible if their exported symbols have the same addresses.

To compare two target shared libraries, we look at their symbols and delete everything except external symbols. Then we create lists of symbol names and values for the new and old libraries, and compare the symbol values to identify differences.

If all symbols in the two libraries have the same values, the libraries are compatible. If some symbols are different, the two libraries may be incompatible. The procedure for comparing shared libraries outlined above is explained in detail in the "Shared Libraries" chapter of the *UNIX System V Programmer's Guide*.

Dealing With Incompatible Libraries

When you determine that two libraries are incompatible, you have to deal with the incompatibility. You can rebuild all the **a.out** files that use your library, or you can give a different target path name to the new version of the library. The host and target path names are independent, so you don't have to change the host library path name. New **a.out** files will use your new target library, but old **a.out** files will continue to access the old library.

You should try to avoid multiple library versions. If too many copies of the same shared library exist, they might actually use more disk space and more memory than the equivalent relocatable version would have.

Extended Software Generation Utilities

The following list describes some of the tools in the Extended Software Generation Utilities package:

- **mcs**(1) is used to manipulate the **.comment** sections in object files. (**.comment** sections are created by **#ident**.) **mcs** can be used to delete, print, compress, or add to **.comment** sections.
- The symbolic debugger, **sdb**(1), is used to examine C language executable files and core files and provides a controlled environment for their execution. When testing C language programs symbolically, breakpoints can be set at executable lines of the source code. These breakpoints force the program to pause at the specified point so that an inspection can be made of the current state of the program.
- The **make**(1) program helps users build and maintain up-to-date versions of programs. **make** simplifies the job of keeping track of which files depend on other files, recently modified files, files that need recompiling after changes, and the sequence of operations needed to make a new version of a program.
- **lex**(1) generates programs to be used in simple lexical analysis of text. **lex** reads a file containing specifications of strings to be matched and associated C code. Whenever the lexical analyzer produced by **lex** matches a specified string in its input, it executes the associated C code.
- **yacc**(1) (Yet Another Compiler-Compiler) is a software tool that accepts an LALR(1) grammar specification and associated C code fragments that represent actions to be taken when a found grammar rule is reduced.

For more information about these commands, see the *UNIX System V Programmer's Guide* and the *UNIX System V Programmer's Reference Manual.*

Source Code Control Utilities

The Source Code Control System (SCCS) can be used to record all enhancements and changes to files, along with comments on each version, to maintain a history of the changes made. Some SCCS functions are

- retrieving any recorded version of a file with comments,
- storing a new version of a file,
- and comparing two versions of an SCCS file.

SCCS takes custody of a file and, when changes are made, identifies and stores them in the file with the original source code and/or documentation. As other changes are made, they too are identified and retained in the file. Each separate set of changes is called a delta. History data can be stored with each version: why the changes were made, who made them, when they were made.

Retrieval of the original or any set of changes is possible. Any version of the file as it develops can be reconstructed for inspection or additional modification.

SCCS Commands

Here is a list of SCCS commands:

get retrieves versions of SCCS files

unget undoes the effect of a **get −e** prior to the file being delta'd

delta applies deltas (changes) to SCCS files and creates new versions

admin initializes SCCS files, manipulates their descriptive text, and controls delta creation rights

prs prints portions of an SCCS file in user specified format

sact prints information about files that are currently out for edit

help gives explanations of error messages

rmdel removes a delta from an SCCS file. Allows removal of deltas created by mistake

cdc changes the commentary associated with a delta

what searches any UNIX System file(s) for all occurrences of a special pattern and prints out what follows it. Useful in finding identifying information inserted by the **get** command

sccsdiff shows differences between any two versions of an SCCS file

comb combines consecutive deltas into one to reduce the size of an SCCS file

val validates an SCCS file

vc a filter that may be used for version control

For instructions on how to use SCCS and detailed descriptions of SCCS commands, see the "Source Code Control System" chapter in the *UNIX System V Programmer's Guide*.

Software Installation Information

You will use the System Administration menu command, **sysadm**, to install the Advanced Programming Utilities on your 3B2 Computer.

Prerequisites

The following paragraphs describe CPLU storage requirements and software dependencies.

Software Dependencies

Before you can install and use APU, you must have installed the Directory and File Management Utilities. Also, if your operating system is UNIX System V Release 3.0 or a later release, you must have installed the System Header Files that came with your operating system.

Issue 1 of APU will be supported on systems running UNIX System V Release 2.0 or later releases. However, the **mkshlib** command will only work on UNIX System V Release 3.0 and later releases, which support the shared library feature.

Storage Requirements

You must meet the following requirements before you begin installation.

- **Memory requirements.** The minimum memory requirement for the APU is 420K of main memory.
- **Storage space.** There must be six megabytes of free disk storage. Installation will fail if there isn't adequate storage space. You can use the **df**(1M) command to check free disk storage.

  You need to have about 250 blocks of free space in your root directory (/), and about 3000 blocks of free storage in **/usr**.

Installing APU

1. Make sure **/usr** is mounted. Type **mount** to see what is mounted. If **/usr** has not been mounted, mount it with the **mount** command

   mount /dev/dsk/c*X*d*Y*s*Z* /usr

   where *X* is replaced by the controller number, *Y* is replaced by the drive number, and *Z* is replaced by the section or slice number where **/usr** is to be mounted.

2. Type the following command line:

   sysadm installpkg

   This executes the system administration subcommand **installpkg**.

3. Insert the first floppy diskette in the Extended Software Generation Utilities set and press RETURN as instructed. After all the utilities on the first diskette have been installed, you will see a message that tells you to remove the first diskette and insert the next one. When you have repeated this procedure for all the diskettes in the package, you will see a message telling you to type **q** to signal the last diskette in the package.

4. Repeat the procedure for the Advanced C Utilities package and, finally, the Source Code Control Utilities package.

Software Notes

This section lists points of interest and workarounds that programmers might need to know about.

1. Functions that use floating point may not be placed in a user-defined shared library. Applications that build their own shared libraries must arrange to place floating point code in a non-shared portion of the host archive shared library.

2. The command **mcs −d** will corrupt a.outs and object files where the comment section is not the last section. Use **mcs −d −ax** instead.

3. When compiling C programs that are the output of **ctrace**, expect to see warning messages of the form:

```
"/usr/lib/ctrace/runtime.c", line nnn warning:
illegal pointer combination, op =
```

4. The following C library functions do not have **lint** library definitions:

mkdir()	**opendir()**	**fpsetmask()**
rmdir()	**readdir()**	**fpsetround()**
sigset()	**closedir()**	**fpsetsticky()**
sighold()	**telldir()**	**isnand()**
sigignore()	**seekdir()**	
sigrelse()		**getutent()**
sigpause()	**lockf()**	**getutid()**
getmsg()	**cfree()**	**getutline()**
putmsg()		**pututline()**
poll()	**fpgetmask()**	**endutent()**
dup2()	**fpgetround()**	**setutent()**
getdents()	**fpgetsticky()**	**utmpname()**

5. The following C library functions have incorrect **lint** library definitions:

setvbuf
signal

6. The files **/usr/options/acu.name** and **/bin/mkshlib** are not removed when the Advanced C Utilities diskette is un-installed.

7. On UNIX System V Release 2.0 systems, **sdb** might look as though it has failed at startup. This is due to the kernel sending a signal to the process when it queries the kernel about its floating point capability. **sdb** reports something like:

```
Bad System Call (12) (sig 12)
 at
fpstart1.c: No such file or directory
0x808????? in sys3b:No lines in file
```

When this message appears, type **c** to continue.

8. In early issues of some C compilation systems, all relocatable object files (**.o** files) produced by the assembler and relocated object files (**a.out** files) produced by the link editor had only three sections: **.text**, **.data**, and **.bss**. However, the assembler in Issue 4 of the C Programming Language Utilities can generate object files with an arbitrary number of sections in an arbitrary order; and the link editor can generate an arbitrary number of sections. This is because of the following features:

   - Addition of a **.comment** section in most object files. (See "**#ident** Preprocessor Directives" in this document.)
   - Elimination of zero-length **.bss** or **.data** sections in **.o** files. This change was introduced to enhance performance of the compilation process.
   - Addition of any number of user-defined sections for special-purpose applications, such as initialization code in some compilation systems.

   Some programs make assumptions about the number of sections in **.o** and **a.out** files. If you use Issue 4 of the C Programming Language Utilities to compile these programs, you should first change the programs so that they read the number of sections in the files. You can do this by reading in the file header using **ldfhread**(3X) and

examining the **f_nscns** field of the file header. See **filehdr**(4) and **ldfhread**(3X) for details.

For example, if you are writing installable device drivers, you should keep in mind that some versions of the **lboot** program (which makes a bootable UNIX System from the kernel and driver modules) assume that the installable drivers have three sections. You need to do two things to your driver's object files before running **lboot**:

- Use the **mcs**(1) command, as shown:

  mcs −d *drivername*.**o**

  This will delete the **.comment** section from the **.o** file.

- Add a **.bss** section to the **.o** file using the **ld** command, as shown:

  ld −r *drivername*.**o −o** *drivername*

This takes *drivername*.**o** and produces the relocatable object *drivername*, attaching an empty **.bss** section to the input file.

If you are using Basic 1.0 and linking **.o** files created from C source files, you should follow this same procedure.

9. You should not use **sdb** to debug any process which uses shared libraries.

10. The following table lists argument/return value types that have changed. (In the second column, the entry "arg2" means "the second argument to the function," "arg3" means "the third argument," etc.)

Function Name	Argument	Changed From	-->	To
fread	**arg2**	**int**	-->	**size_t**
fwrite	**arg2**	**int**	-->	**size_t**
strncat	**arg3**	**int**	-->	**size_t**
strncmp	**arg3**	**int**	-->	**size_t**
strncpy	**arg3**	**int**	-->	**size_t**
ctime	**arg1**	**long**	-->	**time_t**
localtime	**arg1**	**long**	-->	**time_t**
gmtime	**arg1**	**long**	-->	**time_t**

For definitions of the new argument value types, use

#include < sys/types.h >

11. The **mkshlib** command does not accept full pathnames for the **-h** options. It assumes the current directory and prepends the current working directory to the modifier of **-h**.

Documentation

These *APU Release Notes* (select code 307-184) come with APU. The *Release Notes* contain a description of APU and its main features, installation information, prerequisites, and storage requirements.

Related Documents

The following documents contain more information about features of APU and can be ordered as described in the next section.

1. The *C Programming Language Utilities Issue 4 and Advanced Programming Utilities Issue 1 Product Overview* (select code 307-182) contains a brief technical description of the C Programming Language Utilities, Issue 4, and the Advanced Programming Utilities Issue 1. The *Product Overview* is especially useful for new users.
2. The *UNIX System V Programmer's Guide* (select code 307-225) contains descriptive information about SCCS (Source Code Control System), the Link Editor Specification Language, **yacc**, **lex**, **make**, the symbolic debugging program **sdb**, shared libraries, programming on a UNIX System, the C language and associated libraries, the C compiler, and much more.
3. The *UNIX System V Programmer's Reference Manual* (select code 307-226) contains reference material in the form of manual pages for programming commands, system calls, subroutines, libraries, file formats, macro packages, and character-set tables.
4. The *C Programmer's Handbook* (select code 307-135) contains reference material for the C language. Topics covered include syntax, data types, operators and expressions, statements, functions, declarations, program structure, libraries, formatted input/output, and portable C programs.

How to Order Documents

Additional copies of any document or optional documents can be ordered by calling AT&T Customer Information Center (CIC):

1-800-432-6600 (toll free within the continental United States)

1-317-352-8556 (outside the continental United States)

or by writing to:

AT&T Customer Information Center
Customer Service Representative
P. O. Box 19901
Indianapolis, Indiana 46219