Title- **Instructions for GRAPH , TEK , TEKSTARE and GSIP graphics packages**

Date- **June 30, 1976**

TM- **76-1527-35**

Other Keywords- **UNIX,MERT,STARE**

Author	Location	Extension	Charging Case- **39091-7**
A. R. Storm	**MH 1B-124**	**6238**	Filing Case- **39091-7**

ABSTRACT

The **graph** routine provides a grid, scales data and produces a label defining the limits of the abscissa and ordinate. When this information is piped "|" into the routine **tek** it is converted to data compatible with the 4014 storage scope terminal's internal requirements. Similarly, piping into the **gsip** routine produces properly translated and formated data to drive a GSI type terminal in the plot mode. The **tekstare (tks)** routine takes information in the 4014 format, transforms it to STARE compatible information and spawns a job on the HIS 6000 which produces STARE output.

These routines were stolen from the "research" machine with the help of M. D. McIlroy. The **graph** , **tek** and **gsip** routines were written by M. D. McIlroy and L. L. Cherry while the **tekstare (tks)** routine was written by Mike Lesk. The help of all of the above is gratefully acknowledged.

Pages Text 2	Other 7	Total 9	
No. Figures 6	No. Tables 0	No. Refs. 0	

SEE REVERSE SIDE FOR DISTRIBUTION LIST

New Graphic Symbols for EQN and NEQN

Carmela Scrocca

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

There is now available on UNIX and GCOS a set of special characters frequently used in technical typing. In the past, authors have sometimes written out these symbols in English; others just assumed their secretary or typist had these symbols ready and waiting. These characters, however, are not part of the standard terminal or typesetter character sets, but are built-up of those already available. They can presently be produced for phototypesetter output by using EQN/TROFF; NEQN/NROFF can be used for computer terminal output.

This document displays these characters, shows how to use them, and discusses what is involved in making a special character.

Bell Laboratories

subject: UNIX System Call Measurements

date: September 16, 1976

from: C. D. Perez
T. M. Raleigh
MF-76-8234-079

YY-8234-4

MEMORANDUM FOR FILE

Introduction

This memorandum is the first of a series which will deal with fundamental measurements of the UNIX operating system on the PDP-11 line of computers. A description is given of system calls and some of their basic measurements.

Overview

The purpose of the set of measurements described in this memorandum is to explore in detail the overhead incurred in making a system call under UNIX and to obtain some information on the raw processing speed of the current PDP 11 line of processors (11/40, 11/45 and 11/70).

Since the time to execute various system calls is on the order of tens or hundreds of microseconds, a facility for measurement that had a high resolution was needed to obtain the data we desired. A trace tool designed by T.M. Raleigh provided the ability to record time stamped events in a manner similar to the trace facilities available on larger time sharing systems. When used with the KW11-P programmable clock, time stamps accurate to 10 microseconds were obtained. In addition, a data reduction program called fsm was written to reduce the data and produce formatted listings and statistical information.

Using the trace as our basic measurement tool, an environment was created on each of the processors studied that attempted to minimize the number of perturbations of measurements. Unibus interference was the chief problem and this could only be minimized, not eliminated, because of the architecture of the PDP-11.

To conduct the experiment, several system calls were selected which do not require any scarce resources and hence do not need to roadblock until the resource is available. This eliminated any haruware configuration problems such as

1114

Bell Laboratories Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- Synthetic Process for UNIX Date- September 24, 1976

TM- 76-8234-17
76-9156-2

Other Keywords- Performance Evaluation
Workload Measurement
Interprocess Communication

Author D. M. Bernstein
Location MH 2F-245
Extension 2008

Charging Case- 70107-003
Filing Case- 40952-1

ABSTRACT

A synthetic job performs a parameter-specified amount of processor cycles and disk I/O operations. Such jobs have been used successfully in measurement experiments. Patterned after widely publicized versions written in Fortran and PL/1, a UNIX version has been implemented in the C language. Input/output options for this version comprise read, write, getc, putc, getw, putw, as well as messages and pipes. The synthetic job concept has been extended further by providing facilities for issuing an arbitrary sequence of system calls such as fork, exec, kill, nice, sleep and wait. With these facilities, networks of cooperating synthetic processes can be constructed as models of applications. The synthetic process writes self-timing information into a report file. Some measurements of system calls comparing different hardware (PDP-11/45 and /70) and software (UNIX and MERT) are presented for illustration.

Pages Text 6 Other 14 Total 20

No. Figures 1 No. Tables 1 No. Refs. 4

E-1932-U (6-73)SEE REVERSE SIDE FOR DISTRIBUTION LIST

DATE FILE

1115

Bell Laboratories

subject: **NROFF/TROFF Formatting Codes for Departmental Organization Directories on PWB/UNIX**
Case: 39373-99

date: **September 1, 1976**

from: **D. W. Smith**
PY 9141
1B-112 x7315

MEMORANDUM FOR FILE

1. INTRODUCTION

This memo describes a PWB/UNIX documentation facility for departmental organization directories. Keeping these directories on UNIX has the following advantages:

- ease of correction,
- no need to retype an entire chart to make updates,
- ability to move lines within a chart and from one chart to another.

2. MACRO DESCRIPTIONS

The data for a chart is entered into a UNIX file, one file per department. There is a particular two-letter code for each employee classification. Following each two-letter code (a *macro* in UNIX jargon) are the employee's last name and initials, the name the person goes by (first name or nickname), room number, and telephone extension. There are additional macros for the date, department name and number, groups within the department, special symbols which may be added to certain lines, and notes that explain the symbols.

2.1 Codes for Employees

A line for each employee must be typed as follows:

.xx **last-name-and-initials first-name room extension**

where *.xx* is the two-letter code for one of the following occupational classifications:[1]

.DH	Department Head	**.BA**	Business Systems Analyst
.DS	Department Secretary	**.BS**	Business Systems Specialist
.SV	Supervisor	**.GC**	General Clerk
.MT	Member of Technical Staff	**.SC**	Service Clerk
.AM	Associate Member of Technical Staff	**.SS**	Senior Service Clerk
.ST	Senior Technical Associate	**.AI**	Staff Aide
.TA	Technical Associate	**.CO**	Computer Equipment Operator
.LA	Lab Assistant	**.RV**	Resident Visitor
.MG	Member of Administrative Group	**.AS**	Acting Supervisor

In all the macros, one or more blanks are used to separate the different items (the *arguments*) specified for the macro. If blanks are to appear within an argument (e.g., to separate the last name from the initials), the tilde (~) character must be typed in place of the blanks, or the entire argument must be enclosed within double quotes. For example, if Mary Ann Jones is a supervisor in some department, the data line for her could look like:

1. Other codes can be implemented if necessary.

UNOS 1116

Bell Laboratories

Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication (see GEI 13.9-3)

Title – LSI-UNIX System

Date – October 6, 1976

TM – 76-1352-4

Other Keywords – UNIX
Operating Systems
Personal Computer
Intelligent Terminal

Author(s)	Location and Room	Extension
H. Lycklama	MH 7C-211	6170

Charging Case – 39394

Filing Case – 39394-11

ABSTRACT

A modified version of the UNIX operating system has been written to run on the LSI-11 microcomputer with 20K words of primary memory and floppy discs for secondary storage. This configuration permits most of the UNIX user programs to run on the LSI-11 microcomputer. The main programming language used is the structured higher-level language C. A background process as well as foreground processes may be run. A set of subroutines have been written to interface to the file system on the floppy diskettes. Asynchronous read/write routines are also available to the user.

The LSI-UNIX System (LSX) has appeal as a stand-alone system for dedicated applications. It also has many potential uses as an intelligent terminal system. The decreasing costs of hardware make such a system a potential candidate for a very powerful and inexpensive personal computer system.

Pages Text 8	Other 2	Total 10	Address Label
No. Figures 1	No. Tables 0	No. Refs. 11	

E-1932-C (6-73)

SEE REVERSE SIDE FOR DISTRIBUTION LIST

1117

Bell Laboratories

Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication (see GEI 13.9-3)

Title- System for Entering Data Through Computer Displayed Forms

Date- October 22, 1976

TM- 76-1352-6

Other Keywords- LSI-UNIX
CRT Terminal

Author(s)	Location and Room	Extension	Charging Case- 39394
E. W. Stark*			
H. Lycklama (Responsible Engineer)	MH 7C211	6170	Filing Case- 39394-11

ABSTRACT

This document describes two programs which make up a system for managing a database through the use of computer displayed forms. A form entry program displays a form on a CRT screen. The user types into the blanks on the screen and can either look up existing entries on the basis of the filled-in fields, or can store the displayed data as a new entry. In order that the form entry program be useful for a variety of displays, it reads in the display specifications from a file. The form entry program is thus completely independent of the data it is used with.

In order to facilitate the generation of new form displays and their specification files, a form editor program allows the user to manipulate the display directly on the CRT screen where it will be used.

One of the unique features of these programs is their use of program controlled buttons underneath the CRT screen. The buttons are used to perform cursor positioning and special program functions.

The form entry system runs on an LSI-11 computer under the LSX operating system.

*Johns Hopkins University

Pages Text 21 Other 3 Total 24

No. Figures 0 No. Tables 0 No. Refs. 7

Address Label

Title- UNIX-INTELLEC MDS Interface

Date- September 1, 1976

TM- 76-3141-1
76-3142-1

Other Keywords- Intel 8080
Microcomputer Software Support

Author(s)	Location and Room	Extension
Joseph E. Lencoski	WH 3F-314	2921
Stuart A. Tartarone	HO 3G-615	4802

Charging Case- 39484-1

Filing Case- 36692-42

ABSTRACT

The recent introduction of the INTELLEC MDS as a debugging tool for Intel 8080 microcomputer systems coupled with the desirability of using the powerful Intel support software available on UNIX PDP-11 for program development required the design of a hardware/software interface. The software package designed permits programs which have been developed on the UNIX system using the SMAL2 compiler (sc), Intel 8080 assembler (as80), and the link-editor (ld80) to be transmitted over a dialed-up connection in pure binary and loaded into the INTELLEC for execution. This memorandum describes the interface, including a synopsis of the new UNIX commands to access these features, and provides program listings of the new software developed.

Pages Text 4 Other 21 Total 25

No. Figures 4 No. Tables 0 No. Refs. 7

Address Label

111 9

NROFF/TROFF User's Manual

Joseph F. Ossanna

Bell Laboratories
Murray Hill, New Jersey 07974

Introduction

NROFF and TROFF are text processors under the PDP-11 UNIX Time-Sharing System[1] that format text for typewriter-like terminals and for a Graphic Systems phototypesetter, respectively. They accept lines of text interspersed with lines of format control information and format the text into a printable, paginated document having a user-designed style. NROFF and TROFF offer unusual freedom in document styling, including: arbitrary style headers and footers; arbitrary style footnotes; multiple automatic sequence numbering for paragraphs, sections, etc; multiple column output; dynamic font and point-size control; arbitrary horizontal and vertical local motions at any point; and a family of automatic overstriking, bracket construction, and line drawing functions.

NROFF and TROFF are highly compatible with each other and it is almost always possible to prepare input acceptable to both. Conditional input is provided that enables the user to embed input expressly destined for either program. NROFF can prepare output directly for a variety of terminal types and is capable of utilizing the full resolution of each terminal.

Usage

The general form of invoking NROFF (or TROFF) at UNIX command level is

nroff *options files* (or **troff** *options files*)

where *options* represents any of a number of option arguments and *files* represents the list of files containing the document to be formatted. An argument consisting of a single minus (−) is taken to be a file name corresponding to the standard input. If no file names are given input is taken from the standard input. The options, which may appear in any order so long as they appear before the files, are:

Option	*Effect*
−o*list*	Print only pages whose page numbers appear in *list*, which consists of comma-separated numbers and number ranges. A number range has the form *N*−*M* and means pages *N* through *M;* a initial −*N* means from the beginning to page *N;* and a final *N*− means from *N* to the end.
−n*N*	Number first generated page *N*.
−s*N*	Stop every *N* pages. NROFF will halt prior to every *N* pages (default *N*=1) to allow paper loading or changing, and will resume upon receipt of a newline. TROFF will stop the phototypesetter every *N* pages, produce a trailer to allow changing cassettes, and will resume after the phototypesetter START button is pressed.
−m*name*	Prepends the macro file **/usr/lib/tmac.***name* to the input *files*.
−r*aN*	Register *a* (one-character) is set to *N*.
−i	Read standard input after the input files are exhausted.
−q	Invoke the simultaneous input-output mode of the **rd** request.

1120

Bell Laboratories

subject: DOS-BATCH to UNIX Conversions - Case 40979

date: September 1, 1976

from: E. H. Albrecht
5161-760901.01MF

MEMORANDUM FOR FILE

These notes and the associated attachment were written to arm the knowledgeable DOS-BATCH programmer with sufficient background to permit him to develop programs, executable on a "stand alone" PDP 11 processor, using the UNIX operating system. The DOS-BATCH system of Department 5161 is currently used in the mode where the machine is dedicated to one individual at a time. The UNIX environment supports up to 14 simultaneous users. Because UNIX is a time sharing system, it may also be accessed via remote terminals. Therefore, a program may be edited, assembled and linked remotely, providing a suitable terminal is available. Load modules may be transmitted to "stand alone" processors for execution. These arrangements allow for extensive program modifications to be made and tested from the work location, at home or field trial sites. The overall affect of such arrangements, for Department 5161, should be to increase programming flexibility and programmer productivity measurable through an increase in computing resources.

The following highlights various aspects of a DOS to UNIX conversion procedure, and documents my encounters in the conversion process. It is intended to be used more as a directory rather than a tutorial.

DOCUMENTATION

The UNIX operating system is documented in the "UNIX Programmer's Manual" which is divided, via separaters, into many parts. The CONTENTS section of the UNIX Manual briefly describes the commands, functions, etc., contained in the manual. The REFERENCE section of the manual contains several memos, articles, etc., that describe UNIX in greater detail.

The majority of utilities (commands) the beginning programmer will use are located in Section (I) or 1. A command shown in the notation "MKDIR(I)" would indicate that it is to be found in Section 1 or I. Within Section (I), commands are ordered alphabetically.

1121

PWB/UNIX Documentation Roadmap

J. R. Mashey

Bell Laboratories
Piscataway, New Jersey 08854

1. INTRODUCTION

A great deal of documentation exists for PWB/UNIX. It has different formats, is contributed by many different people, and is modified frequently. New users are often overcome by the volume and distributed nature of the documentation. This "roadmap" attempts to be a terse, up-to-date outline of crucial documents and information sources.

Numerous people have contributed comments and information for this "roadmap," in order to make it as helpful as possible for PWB/UNIX users. *However, many of these comments are accurate only with regard to PWB/UNIX and may well be totally inapplicable to other versions of UNIX.*

1.1 Things to Do

See a local PWB/UNIX system administrator to obtain a "login name" and get other appropriate system information.

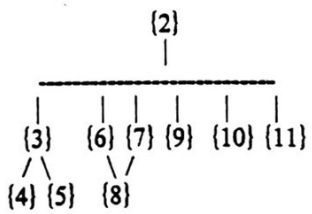
1.2 Notation Used in this Roadmap

{*N*} — Section N in this "roadmap."
++ — item required for everyone.
\+ — item recommended for most users.

All other items are optional and depend on specific interest (a list of relevant documents appears in the Table of Contents of *Documents for the PWB/UNIX Time-Sharing System*).

Items in Section *N* of the *PWB/UNIX User's Manual* are referred to by *name*(N).

1.3 Prerequisite Structure of Following Sections



2. BASIC INFORMATION

Don't do anything else until you have learned most of this section. You must know how to log onto the system, make your terminal work correctly, enter and edit files, and perform basic operations on directories and files.

2.1 PWB/UNIX User's Manual ++

- Read *Introduction* and *How to Get Started.*
- Look through Section I to become familiar with command names.
- Note the existence of the *Table of Contents* and of the *Permuted Index.*

Section I will be especially needed for reference use.

2.2 UNIX for Beginners ++

2.3 A Tutorial Introduction to the UNIX Text Editor ++

122

Bell Laboratories Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **A Real-Time Time-shared Operating System for an SEL/86 - PDP-11 Configuration**

Date- **October 29, 1976**

TM- **76-8231-9**
76-1228-11

Other Keywords- UNIX
MERT

Author	Location	Extension	Charging Case- 38794
W. A. Burnette	MH 2D-233	x4812	Filing Case- 38794-23

ABSTRACT

A new operating system has been written for the SEL/86 - PDP-11 computer configuration in Department 1228. The primary purpose of the SEL - PDP-11 system is to provide real-time interactive computing for research in speech and graphics. The previous operating systems on these two computers supported a single real-time user in the same style as the DDP-224 computers in Dept. 1228. These operating systems have now been replaced by multi-user systems for more effective utilization of both the SEL and the PDP-11.

In this SEL/PDP-11 configuration, the SEL is connected via a high speed channel to the PDP-11 which controls several mass storage devices and provides the mass storage services for the SEL. The SEL controls a number of acoustic, graphical, and interactive devices.

For the PDP-11, the MERT operating system was adopted. System-level programs were written to provide communication between the SEL and the MERT file system. Simultaneously, MERT provides UNIX timesharing services. Thus users can access the same file system from either the PDP-11 or the SEL.

For the SEL, a new operating system was written with a foreground-background scheduler. The foreground supports the real-time user with uninterrupted processing until the program needs to wait for input from interactive devices. Background jobs are processed only while the foreground is waiting for user input. Background jobs may be prepared on remote terminals, using MERT/UNIX, and queued until the SEL is available for background processing.

In this way, the new SEL/PDP-11 system retains the single-user services for which this computer system was intended, but also provides a UNIX-style file system common to both machines, a UNIX time-shared program development facility, and a remote job execution and debugging capability for SEL programs.

Pages Text 12	Other 1	Total 13	
No. Figures 0	No. Tables 0	No. Refs. 7	

SEE REVERSE SIDE FOR DISTRIBUTION LIST

1123

MINI-UNIX Summary

A. Hardware

MINI-UNIX runs on a DEC PDP11/10, 11/20 or 11/40 with at least the following equipment:

28K words of memory: parity not used,
disk: RK05(preferably 2) or equivalent,
console typewriter,
clock: KW11-L or KW11-P,

The system is normally distributed on 9-track tape or RK05 packs.
The following equipment may be supported:

communications controllers such as DL11, DC11 or DH11,
full duplex 96-character ASCII terminals,
RP03, RP04 disks,
9-track tape, or extra disk for system backup.

The memory and disk space specified is enough to run and maintain MINI-UNIX. More will be needed to keep all source on line, or to handle a large number of users, big data bases, diversified complements of devices, or large programs. MINI-UNIX does swapping to provide multi-programming support. The resident code of MINI-UNIX occupies 12-16K depending on configuration. The system as distributed occupies 12K words of memory, allowing 16K words of memory for the user programs. Some editing of source code is required to add new disk, tape or communication device drivers, as the system size will likely expand beyond 12K words. Keep in mind that the C compiler requires a minimum of 12K words of memory to run.

An 11/10, 11/20 or 11/40 is not advisable for heavy floating point work, as MINI-UNIX on this hardware uses interpreted 11/45 floating point.

B. Software

All the programs available as MINI-UNIX commands are listed. Every command, including all options, is issued as just one line, unless specifically noted as "interactive". Interactive programs can be made to run from a prepared script simply by redirecting input.

File processing commands that go from standard input to standard output are noted as usable as filters. The piping facility of the Shell may be used to connect filters directly to the input or output of other programs. Note: pipes are not supported in the system as in standard UNIX. However "pseudo-pipes" are supported in the Shell to enable using commands as filters.

Software listed in Section 6, "Typesetting", is distributed separately as an enhancement to MINI-UNIX. Source code is included except as noted.

1 Basic Software

This package includes the time-sharing operating system with utilities, a machine language assembler and a compiler for the programming language C—enough software to write and run new applications and to maintain or modify MINI-UNIX itself.

1.1 Operating System

= MINI-UNIX The basic resident code on which everything else depends. Supports the system calls, and maintains the file system. A general description of UNIX design philosophy and system facilities appeared in the Communications of the ACM, July, 1974. The following UNIX capabilities are *not* included in MINI-UNIX:

- Separate instruction and data spaces on 11/45 and 11/70.
- "Group" access permissions for cooperative projects, with overlapping memberships.

1124

SETTING UP MINI-UNIX — Sixth Edition

Enclosed are:

1. 'UNIX Programmer's Manual,' Sixth Edition.
2. Documents with the following titles:

   Setting Up MINI-UNIX — Sixth Edition
   The UNIX Time-Sharing System
   C Reference Manual
   Programming in C — A Tutorial
   UNIX Assembler Reference Manual
   A Tutorial Introduction to the UNIX Text Editor
   UNIX for Beginners
   RATFOR — A Preprocessor for a Rational Fortran
   YACC — Yet Another Compiler-Compiler
   NROFF Users' Manual
   The UNIX I/O System
   A Manual for the Tmg Compiler-writing Language
   On the Security of UNIX
   The M6 Macro Processor
   A System for Typesetting Mathematics
   DC — An Interactive Desk Calculator
   BC — An Arbitrary Precision Desk-Calculator Language
   The Portable C Library (on UNIX)
   MINI-UNIX Summary
   Regenerating System Software

3. The MINI-UNIX software on magtape or disk pack.

If you are set up to do it, it might be a good idea immediately to make a copy of the disk or tape to guard against disaster. The tape contains 12100 512-byte records followed by a single file mark; only the first 4000 512-byte blocks on the disk are significant.

The system as distributed corresponds to three fairly full RK packs. The first contains the binary version of all programs, and the source for the operating system itself; the second contains all remaining source programs; the third contains manuals intended to be printed using the formatting programs roff or nroff. The 'binary' disk is enough to run the system, but you will almost certainly want to modify some source programs.

Making a Disk From Tape

If your system is on magtape, perform the following bootstrap procedure to obtain a disk with the binaries.

1. Mount magtape on drive 0 at load point.
2. Mount formatted disk pack on drive 0.
3. Key in and execute at 100000

1145

Bell Laboratories

subject: DEC Factory Acceptance Test

date: January 19, 1977

from: R. B. Brandt
C. D. Perez
MF-77-8234-001

MEMORANDUM FOR FILE

1. Introduction

During June and July, 1976, meetings were held between members of BTL Division 52 and representatives of Digital Equipment Corporation (DEC). [1,2] The primary purpose of these meetings was to discuss some of the problems encountered by Operations Support Systems (OSS's) in the field, especially those resulting in long outages. A number of proposals for improving the installation and maintenance of these OSS's were advanced. One such proposal was that BTL should make available to DEC a UNIX checkout package for use as a final factory acceptance test on DEC-supplied OSS hardware prior to shipment to the field site. As a result of this suggestion, it was decided that such a checkout package would be developed and used on a trial basis for approximately six months on Switching Control Center System (SCCS) configurations. This memorandum is a description of the checkout package that is being used in the trial.

2. Overview

The package that was provided to DEC in early December, 1976, consists of a UNIX "sysgen" tape and the documentation describing how to boot UNIX from the tape and run the tests. The "sysgen" tape is like those supplied by the UNIX Support Group (USG) to BTL projects; it contains a root file system image and the programs that are necessary to copy the tape to a disk and boot the UNIX system. Source code was not included, and the UNIX on the tape was configured for SCCS (11/70 with 128K memory, TU16, RP05, and four DH11's) at modification level 2.0j. A copy of the documentation that accompanied the "sysgen" tape is attached to this memorandum (Attachment A). The checkout system is to be run on all SCCS machines after all other DEC factory acceptance tests have been completed. Test results are to be documented so that an evaluation of the trial may be made at its completion. The checkout package is not supplied to the field site for use during actual system installation. See Reference [3] for a breakdown of test responsibilities for DEC and BTL.

Before proceeding with a detailed description of the tests

1125

A Tutorial Introduction to ADB

J. F. Maranzano

S. R. Bourne

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Debugging tools generally provide a wealth of information about the inner workings of programs. These tools have been available on UNIX† to allow users to examine "core" files that result from aborted programs. A new debugging program, ADB, provides enhanced capabilities to examine "core" and other program files in a variety of formats, run programs with embedded breakpoints and patch files.

ADB is an indispensable but complex tool for debugging crashed systems and/or programs. This document provides an introduction to ADB with examples of its use. It explains the various formatting options, techniques for debugging C programs, examples of printing file system information and patching.

May 5, 1977

†UNIX is a Trademark of Bell Laboratories.

PWB/UNIX

Shell Tutorial

J. R. Mashey

September 1977

Bell Telephone Laboratories, Incorporated

1194

1/80

Setting Up PWB/UNIX

R. C. Haight

Bell Laboratories
Murray Hill, New Jersey 07974

M. J. Petrella

Bell Laboratories
Piscataway, New Jersey 08854

L. A. Wehr

Bell Laboratories
Murray Hill, New Jersey 07974

1. INTRODUCTION

1.1 Prerequisites

Before attempting to generate a PWB/UNIX* system, you should understand that a considerable knowledge of the attendant documentation is required and assumed. In particular, you should be very familiar with the following documents:

- *The UNIX Time-Sharing System*
- *PWB/UNIX User's Manual*
- *C Reference Manual*
- *Administrative Advice for UNIX/TS*
- *PWB/UNIX Operations Manual*

A complete list of pertinent documentation is contained in *Documents for PWB/UNIX*. Throughout this document, each reference of the form *name*(N), where N is a number, refers to entry *name* in Section N of the *PWB/UNIX User's Manual*.

You must have a basic understanding of the operation of the hardware. This includes the console panel, the tape drives, and the disk drives, all of which are assumed to have standard addresses and interrupt vectors. It is also assumed that the hardware works and has been completely installed. The DEC® diagnostics should have been run to test the configuration, and you must have a detailed description of the hardware, including device addresses, interrupt vectors, and bus levels. This information is very important to generate the PWB/UNIX system.

Older versions of PWB/UNIX cannot be correctly *updated* with a PWB/UNIX Release 2.0 system; therefore, the installation of PWB/UNIX Release 2.0 must be done as an initial load.

1.2 Procedure

PWB/UNIX is distributed on two magnetic tapes, recorded in 9-track format at 800-bpi. Tape 1 is essential to the entire procedure, because it contains the initial load program and a copy of the *root* file system, as well as a copy of the */usr* file system (which contains source and supplemental commands). The initial load program will copy the *root* file system from tape (either a TU10 or a TU16) to disk (either an RP03 or an RP06). Note that RP04 and RP05 drives are considered to be equivalent to RP06 drives; any differences will be noted explicitly. Once the *root* file system has been successfully loaded to disk, PWB/UNIX may be booted and the available utility programs may then be used to complete the installation. The */usr* file system contains information essential to generating a new system that will match your particular hardware and software environment.

Tape 2 contains the machine-readable manual pages, as well as the source for selectable subsystems of PWB/UNIX.

---

* UNIX is a Trademark of Bell Laboratories.

1195
1/80

Administrative Advice for UNIX/TS

R. C. Haight

Bell Laboratories
Murray Hill, New Jersey 07974

The material presented here is based on the author's experiences and opinions. Nevertheless, it may prove useful. The material on phototypesetting was contributed by D. W. Smith.

1. ADMINISTRATOR'S ROAD MAP

Getting started as a UNIX†/TS system administrator is hard work. There are no real shortcuts to a working knowledge of the system. You will need time for reading, study and hands-on experimenting. Don't commit yourself to "going live" with your system until you have had two weeks to teach yourself your job, and get the initial hardware quirks ironed-out.

Don't consign the *Setting Up UNIX/TS* document to oblivion after your initial system "gen". In addition to needing it again whenever you add/change equipment, you will find that it contains valuable material about system tuning (buffers, *clist*s, etc.) that appears nowhere else.

As an administrator, you should be familiar with a lot of the distributed documentation. The *Internals, Operations, and Administration* papers from *Documents for UNIX/TS* should all be studied, as well as the *Introduction*, *How to Get Started*, and most of the entries of the *UNIX/TS User's Manual*. In that manual, you should pay special attention to: *acct**(1M)*, *chmod*(1), *chown*(1), *config*(1M), *cpio*(1), *date*(1), *df*(1), *du*(1), *ed*(1), *env*(1), *find*(1), *fsck*(1M), *kill*(1), *mail*(1), *mkdir*(1), *mkfs*(1M), *ncheck*(1M), *ps*(1), *rm*(1), *rmdir*(1), *shutdown*(1M), *stty*(1), *su*(1), *sync*(1M), *time*(1), *volcopy*(1M), *wall*(1M), *who*(1), and *write*(1) in Section 1; all of Section 4; *acct*(5) in Section 5; and *crash*(8) and *vaxops*(8) in Section 8.

2. SYSTEM CAPACITY

The figures below are approximations based on our experience over several years:

Hardware Configuration	*Number of Simultaneous Users*
PDP-11/23; 256K-byte memory; 2 RL01 disks*	4
PDP-11/34; 256K-byte memory with cache; 2 RL01 disks*	8
PDP-11/45; 248K-byte memory; RP03 disk*	16
Above with RP06 (RP04, RP05) disk*	20
Above with memory cache	25
PDP-11/70; 512K-byte memory; RP06 (RP04, RP05) disks* (2 or more drives)	32
Above with 768K-byte memory and a disk drive (or fixed-head disk) set aside for the root file system	40
VAX-11/780; 1M-byte memory; at least 3 RP06 disks*	48

* Or equivalent.

See *Setting Up UNIX/TS* for the list of supported hardware options.

† UNIX is a Trademark of Bell Laboratories.

1196

1/80

PWB/UNIX Operations Manual (Second Edition)

A. G. Petruccelli

Bell Laboratories
Piscataway, New Jersey 08854

ABSTRACT

This manual contains a complete description of PDP® 11 console operations, step-by-step instructions for normal operator functions, as well as descriptions of the PWB/UNIX system console error messages.

The information in this manual was gathered from personal experience, the *PWB/UNIX User's Manual*, Digital Equipment Corporation (DEC®) hardware manuals, and technical memorandums contained in *Documents for PWB/UNIX*.

Because this manual is intended to be as general as possible, it is suggested that each location add specific information about:

- Hardware configuration.
- Telephone line configuration.
- Specific logging and record-keeping practices.
- Contacts for hardware and software problems.
- Site-dependent diagnostic procedures.

This manual is partially based on, and supercedes the *PWB/UNIX Operations Manual* by M. E. Pearlman.

Repairing Damaged PWB/UNIX File Systems

P. D. Wandzilak

March 1978

Bell Telephone Laboratories, Incorporated

1234

Bell Laboratories Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication (see GEI 13.9-3)

Title: **Interprocess Communication Mechanisms in CB-UNIX**

Date: **November 21, 1977**

TM: **77-5223-1**
5223-771121.01TM

Other Keywords: **UNIX File System**
Pipes
Named Pipes
Signals
Semaphores
MAUS
Messages

Author(s)	Location	Extension	Charging Case: 49359-20
J. C. Kaufeld Jr.	**CB 2C-249**	**4522**	Filing Case: **49075-01**

ABSTRACT

A discussion of interprocess communication mechanisms in CB-UNIX[1] is presented. In particular: files, pipes, named pipes, signals, semaphores, MAUS and messages are discussed. For each mechanism, a general explanation is given, the user interface is detailed and then limitations and potential problems are presented. The discussion applies specifically to CB-UNIX, a version of the UNIX operating system developed in Columbus for use in real-time oriented applications. The explanations do not necessarily apply to UNIX operating systems in general, however, most versions of the UNIX operating system have very similar implementations of files, pipes, signals and messages.

ERRATTA

This memo is a corrected copy issued June 19, 1978. This copy corrects a number of misleading comments and outright errors. In addition, references to the CB-UNIX programmers manual are corrected to refer to the correct section number.

Pages Text: 22	Other: 1	Total: 23	
No. Figures: 0	No. Tables: 0	No. Refs.: 5	

1239

An Introduction to the UNIX Shell

S. R. Bourne

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The *shell* is a command programming language that provides an interface to the UNIX† operating system. Its features include control-flow primitives, parameter passing, variables and string substitution. Constructs such as *while, if then else, case* and *for* are available. Two-way communication is possible between the *shell* and commands. String-valued parameters, typically file names or flags, may be passed to a command. A return code is set by commands that may be used to determine control-flow, and the standard output from a command may be used as shell input.

The *shell* can modify the environment in which commands run. Input and output can be redirected to files, and processes that communicate through 'pipes' can be invoked. Commands are found by searching directories in the file system in a sequence that can be defined by the user. Commands can be read either from the terminal or from a file, which allows command procedures to be stored for later use.

November 12, 1978

†UNIX is a Trademark of Bell Laboratories.

1246

Bell Laboratories Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **The MERT Operating System**

Date- **March 22,1978**

TM- **78-3114-3**
78-1352-3

Other Keywords- **UNIX**
Real-Time
Multi-Environment
Interprocess Communication

Author	Location	Extension	Charging Case- 39394
H. Lycklama	HO 1G-317	3212	Filing Case- 39394-11
D. L. Bayer	MH 7C-207	3080	

ABSTRACT

The MERT operating system[+] supports multiple operating system environments. Messages provide the major means of inter-process communication. Shared memory is used where tighter coupling between processes was desired. The file system was designed with real-time response being a major concern. The system has been implemented on the DEC PDP-11/45 and PDP-11/70 computers and supports the UNIX time-sharing system as well as some real-time processes.

The system is structured in four layers. The lowest layer, the *kernel*, provides basic services such as inter-process communication, process dispatching, and trap and interrupt handling. The second layer comprises privileged processes, such as I/O device handlers, the file manager, memory manager, and system scheduler. At the third layer, the supervisor processes provide the programming environments for application programs of the fourth layer. To provide an environment favorable to applications with real time response requirements, processes are permitted to control scheduling parameters such as scheduling priority and memory residency. A rich set of inter-process communication mechanisms including messages, events (software interrupts), shared memory, inter-process traps, process ports, and files, allow applications to be implemented as several independent, cooperating processes.

Some uses of the MERT operating system are discussed. A retrospective view of the MERT system is also offered. This includes a critical evaluation of some of the design decisions and a discussion of design improvements which could have been made to improve overall efficiency.

[+] A modified version of this memorandum was submitted for publication in the special issue of the BSTJ on Software, July-August, 1978.

UNIX is a Trademark of Bell Laboratories.

Pages Text 23	Other 5	Total 28
No. Figures 4	No. Tables 0	No. Refs. 13

E-1932-U (6-73)

SEE REVERSE SIDE FOR DISTRIBUTION LIST

1247

Bell Laboratories Cover Sheet for Technical Memorandum

Title- **A Minicomputer Satellite Processor System +**

Date- **March 22,1978**

TM- **78-3114-2**
78-1359-3

Other Keywords- **UNIX**
Operating Systems
Minicomputer Support
Microprocessors

Author	Location	Extension	Charging Case- **39394**
H. Lycklama	**HO 1G-317**	**3212**	Filing Case- **39394-11**
C. Christensen	**MH 7C-217**	**4441**	

ABSTRACT

A software support system + for a network of minicomputers and microcomputers is described. A powerful time-sharing system on a central computer controls the loading, running, debugging and dumping of programs in the satellite processors. The fundamental concept involved in supporting these satellite processors is the extension of the central processor operating system to each satellite processor. Software interfaces permit a program in the satellite processor to behave as if it were running in the central processor. Thus, the satellite processor has access to the central processor's I/O devices and file system yet has no resident operating system. The implementation of this system was considerably simplified by the fact that all processors, central and satellite, belong to the same family of computers (DEC PDP-11 series). We describe some examples of how the SPS system is used in various projects at Bell Laboratories.

+ A modified version of this memorandum has been submitted for publication in the special issue of the BSTJ on Software, July-August, 1978.

Pages Text 7	Other 3	Total 10	
No. Figures 2	No. Tables 0	No. Refs. 8	

SEE REVERSE SIDE FOR DISTRIBUTION LIST

1291
UNSD

Bell Laboratories

Subject: Recommendations for A Company-Wide UNIX* Education Program

date: June 6, 1978

from: R. M. Coben
E. H. Dudley
R. A. Faulkner
R. F. Graveman
B. W. Kernighan
R. D. Reese
R. F. Rosin
D. S. Sand
G. C. Vogel
G. A. Wilson
B. C. Wonsiewicz

ABSTRACT

Bell Laboratories has become heavily committed to the UNIX time-sharing system, its associated utilities, and the programming language C. Although current efforts in UNIX education are extensive, they tend to be disorganized, redundant, and incomplete. The Ad Hoc UNIX Education Committee was chartered to recommend a solution to this problem, addressing specifically the needs of the technical population.

Our committee recommends that a new supervisory group be formed whose sole concern will be the many aspects of UNIX education. This supervisory group should be aligned with a central UNIX system support organization.

This report presents an overview of the use of UNIX at Bell Laboratories, our view of the challenge for UNIX education, and our specific recommendations. Appendices to this report cover these topics in detail and include tentative recommendations for course modules and outlines.

*UNIX is a Trademark of Bell Laboratories

1302
UNDS

Bell Laboratories

subject: Exercises in Repairing PWB/UNIX File Systems
File: 39382-900

date: October 19, 1978

from: P. D. Wandzilak
PY 9442
1A-114 x7344

MEMORANDUM FOR FILE

1. INTRODUCTION

This document describes a tool that generates a series of self-instructional exercises based on the concepts presented in reference [3]. The tool is an interactive PWB/UNIX* program that simulates the various degrees of file structure damage that are presently reported by the check(VIII) command. The workshop exercises are intended to provide users with the necessary framework that will enable them to become more proficient in repairing a damaged PWB/UNIX file system.

I assume that the reader has an understanding of the descriptions in references [1], [2], and [3]. I also recommend that copies of this document and reference [3] be periodically consulted for detailed explanations and procedures on resolving the more complicated exercises that might be generated in using this tool.

2. BASIC CAPABILITIES

Fsrepair is an interactive PWB/UNIX program that selectively alters the structure of a consistent file system to contain one or more of the various degrees of file structure damage that are currently reported by check. The tool allows the individual to concentrate on the problem area(s) to which he or she needs more exposure. Figure 1 contains a complete list of all the file system damage that is generated by this program.

A. Block Diagnostics

Mnemonic Identifier	Description
bad	The block number contains a value outside the allowable space on the file system.

* UNIX is a Trademark of Bell Laboratories.

UNOS 1320

Bell Laboratories

subject: Setting Up UNIX/TS

date: September 30, 1978

from: R. C. Haight
MH 8234
2F211 x7498
MF 78-8234-98

L. A. Wehr
MH 8234
2F245 x4896
MF 78-8234-98

MEMORANDUM FOR FILE

The attached document describes programming steps for generating a UNIX/TS operating system along with administrative detail on configuration, setting up file systems, and installation/recompilation of command software.

R. C. Haight

MH-8234-RCH/LAW

L. A. Wehr

UNOS 1335

Bell Laboratories **Cover Sheet for Technical Memorandum**

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **UNIX File Security**

Date- **January 19, 1979**

TM- **79-1271-3**

Other Keywords- **Privacy**
Encryption
Secret Writing

Author	Location	Extension	Charging Case- 39199
Robert Morris	**MH 2C524**	**3878**	Filing Case- **39199-11**

ABSTRACT

This paper describes the history of the design of the file encryption scheme on the UNIX time-sharing system. The present design was the result of countering observed attempts to penetrate the system. The result is a well-understood scheme that is fast and easy to use.

Pages Text 6	Other 1	Total 7
No. Figures 0	No. Tables 0	No. Refs. 10

SEE REVERSE SIDE FOR DISTRIBUTION LIST

UNOS 1345

Bell Laboratories **Cover Sheet for Technical Memorandum**

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **An Essay in Computer Security: Decrypting A Former UNIX† *crypt***

Date- **December 29, 1978**

TM- **78-1271-20**

Other Keywords- **Ciphers**

Author	Location	Extension	Charging Case- **39199**
P.J. Weinberger	**MH2C-514**	**7214**	Filing Case- **39199-11**

ABSTRACT

Users of computer systems need some way of keeping their files private from non-privileged users sharing the same machine. On UNIX at least, relying on the file protection mechanism decreases the amount of legitimate sharing which is possible, and leaves the file readable on dump tapes. A series of encryption schemes have been tried on the Center 127 system. The next-to-last of these, used from March to December of 1978, seems to be quite satisfactory for small files However, the techniques discussed in this memo generally recover at least 70% of the plain text of the file given about 11000 characters of enciphered text. The cryptanalysis algorithm takes about two minutes of processor time. Another 10 minutes of human intervention usually completely decrypts the text.

†UNIX is a Trademark of Bell Laboratories.

TM-78-1271-20

Pages Text 5	Other 0	Total 5
No. Figures 0	No. Tables 0	No. Refs. 0

BIREN,IRMA B
MH2F128
SUBJECT MATCH

CM
01/31/79
COCSFS

E-1932-U (6-73) SEE REVERSE SIDE FOR DISTRIBUTION LIST

MH

1353
UNPL

Bell Laboratories **Cover Sheet for Technical Memorandum**

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **Semantics of the C programming language, part 0: prelude**

Date- **January 2, 1979**

TM- **79-1271-2**

Other Keywords-

Author	Location	Extension	Charging Case- **39199**
Ravi Sethi	**MH 2C-519**	**4006**	Filing Case- **39199-11**

ABSTRACT

This is the first of a sequence of papers defining the semantics of the C programming language. Of the three methods — operational, denotational, and axiomatic — that have been used to specify the semantics of reasonably complete languages, the denotational method has been chosen to specify C. In this prelude, a very simple language with assignments and while loops will be used to illustrate the semantic method.

Pages Text 23	Other 7	Total 30	WORKING COPY
No. Figures 10	No. Tables 0	No. Refs. 59	

E-1932-U (6-73) SEE REVERSE SIDE FOR DISTRIBUTION LIST

UNOS 1358

Bell Laboratories

subject: UNIX Command Syntax

Filing Case 40125-001

date: February 16, 1979

from: A. S. Cohen
MH 2524
2F-223 x6920

S. B. Olsson
MH 2524
2C-253 x3474

G. C. Vogel
MH 2524
2C-158 x6115

2524-79-0216-02MF

ABSTRACT

The current UNIX™ command-line syntax is riddled with inconsistencies. Command-syntax rules and a library routine for achieving consistent syntax are proposed.

MEMORANDUM FOR FILE

There are now thousands of Bell Labs employees using UNIX™ time sharing systems. Many of these people are occasional users who are unfamiliar with the internal workings of the system, the many features of the shell, the historical development of the system, or the quirks of the command syntax.

Consider the following command lines:

```
lpr −l −m wlist
lpr −cm zlist
tp tm1 gold
pr −h "Grocery List" −5 −w100 glist
```

A user might reason from the first two lines that options may be grouped for convenience. The third line suggests a command syntax in which the command modifiers (options or keys) need not be prefixed by a '−'. Finally, the last line displays a mixture of options taking arguments. The 'h' option allows white space before its argument, the 'w' option allows its argument to be adjacent, and the '−' takes its argument without white space. One is apt to feel that, in spite of the variety, the command syntax is quite liberal.

1367
1180

Source Code Control System

User's Guide

L. E. Bonanni
C. A. Salemi

Bell Telephone Laboratories, Incorporated

UNOS 1369

Sdb: A Symbolic Debugger

Howard P. Katseff

Bell Laboratories
Holmdel, New Jersey 07733

1. Introduction

This document describes a symbolic debugger, *sdb*, as implemented for C and F77 programs on the UNIX/32V† Operating System. *Sdb* is useful both for examining core images of aborted programs and for providing an environment in which execution of a program can be monitored and controlled.

2. Examining core images

In order to use *sdb*, it is necessary to compile the source program with the '−g' flag. This causes the compiler to generate additional information about the variables and statements of the compiled program. When the debug flag is specified, *sdb* can be used to obtain a trace of the called procedures at the time of the abort and interactively display the values of variables.

2.1. Invoking sdb

A typical sequence of shell commands for debugging a core image is:

```
% cc −g foo.c −o foo
% foo
Bus error − core dumped
% sdb foo
main:25:      x[i] = 0;
*
```

The program foo was compiled with the '−g' flag and then executed. An error occurred which caused a core dump. *Sdb* is then invoked to examine the core dump to determine the cause of the error. It reports that the Bus error occurred in procedure main at line 25 (line numbers are always relative to the beginning of the file) and outputs the source text of the offending line. *Sdb* then prompts the user with a '*' indicating that it awaits a command.

It is useful to know that *sdb* has a notion of current procedure and current line. In this example, they are initially set to 'main' and '25' respectively.

In the above example *sdb* was called with one argument, 'foo'. In general it takes three arguments on the command line. The first is the name of the executable file which is to be debugged; It defaults to a.out when not specified. The second is the name of the core file, defaulting to core and the third is the name of the directory containing the source of the program being debugged. *Sdb* currently requires all source to reside in a single directory. The default is the working directory. In the example the second and third arguments defaulted to the correct values, so only the first was specified.

It is possible that the error occurred in a procedure which was not compiled with the debug flag. In this case, *sdb* prints the procedure name and the address at which the error occurred. The current line and procedure are set to the first line in main. *Sdb* will complain if main was not compiled with '−g' but debugging can continue for those routines compiled with

†UNIX is a trademark of Bell Laboratories

UNDS 1575

Bell Laboratories

subject: Using UNIX Capabilities More Effectively.
File: 38957-32

date: Jan. 30, 1979

from: W. J. Mayer
HO 5113
20624 x2306
5113-790130.01MF

ABSTRACT

A new UNIX program is introduced in this memo that allows convenient transfer of data between UNIX and non-UNIX time shared systems. Although standard UNIX software provides communications with other UNIX systems and with some other BTL batch processors, direct data transfers with non-UNIX time shared systems have not previously been possible in a suitably convenient manner.

This new program makes it feasible to complement the capabilities of other systems with the capabilities that are unique to UNIX. UNIX can now be used to prepare source data for another system, run a program with the prepared input and have the output sent back to UNIX for further processing. Or, UNIX can act as the "middle man" to transfer data, with or without editing, between two other systems that may have no other convenient means of communications.

The program is currently being used as a link between EISS (Economic Impact Study System) and the IBM batch processing (through UNIX) and as a link between UNIX and TASP (Toll Alternatives Studies Program) at the AT&T, Piscataway computer facility.

Copy to
(Without att. III)
All Supervision Center 511
W. J. Bebio
J. R. Hackett
T. T. Lee
R. A. Norden
N. M. Rees
T. L. Russell

UNPL 1376

Bell Laboratories **Cover Sheet for Technical Memorandum**

Title- **Semantics of the C programming language, part 1: statements**

Date- **February 2, 1979**

TM- **79-1271-04**

Other Keywords-

Author	Location	Extension	Charging Case- 39199
Ravi Sethi	**MH 2C-519**	**4006**	Filing Case- 39199-11

ABSTRACT

This is one of a sequence of papers defining the semantics of the C programming language. After providing a brief introduction to the semantic method, the semantics of statements are given in section 3. Section 3 was prepared by adding denotational semantics for the various statements to the discussion in section 9 of the C Reference Manual, as it appears in: B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ, 1978. The work reported here was carried out with the cooperation of F. T. Grampp and A. R. Koenig.

Pages Text	16	Other	10	Total	26
No. Figures	4	No. Tables	0	No. Refs.	9

Carole Schaderman
MH 2F-128