NEW REAL-TIME OPERATING SYSTEMS FOR EMBEDDED SYSTEMS

NEW REAL-TIME OPERATING SYSTEMS FOR EMBEDDED SYSTEMS

By

K. SRINIVASA REDDY

M.Tech (Ph.D.)

Associate Prof. Electronics and Communication Engineering

Nagole Institute of Technology and Science

Hyderabad

Andhra Pradesh

UNIVERSITY SCIENCE PRESS

(An Imprint of Laxmi Publications Pvt. Ltd.)

BANGALORE • CHENNAI • COCHIN • GUWAHATI • HYDERABAD
JALANDHAR • KOLKATA • LUCKNOW • MUMBAI • PATNA

RANCHI • NEW DELHI

Published by

UNIVERSITY SCIENCE PRESS

(An Imprint of Laxmi Publications Pvt. Ltd.)

113, Golden House, Daryaganj,
New Delhi-110002

Phone : 011-43 53 25 00
Fax : 011-43 53 25 28

www.laxmipublications.com
info@laxmipublications.com

Price : ₹ **95.00** *Only.* *First Edition :* 2013

OFFICES

✆ **Bangalore**	080-26 75 69 30	✆ **Chennai**	044-24 34 47 26
✆ **Cochin**	0484-237 70 04, 405 13 03	✆ **Guwahati**	0361-251 36 69, 251 38 81
✆ **Hyderabad**	040-24 65 23 33	✆ **Jalandhar**	0181-222 12 72
✆ **Kolkata**	033-22 27 43 84	✆ **Lucknow**	0522-220 99 16
✆ **Mumbai**	022-24 91 54 15, 24 92 78 69	✆ **Patna**	0612-230 00 97
✆ **Ranchi**	0651-221 47 64		

UNR–9662–095–NEW REAL-TIME OPER SYS-RED **C—**

Typeset at ABRO Enterprises, Delhi. *Printed at*

PREFACE TO THE FIRST EDITION

An embedded operating system (OS) or Real-Time Operating System (RTOS) is a multitasking operating system intended for real-time applications and designed to be very compact and efficient, forsaking many functions that non-embedded computer operating systems provide. An embedded OS or RTOS provides facilities which, if used properly, guarantee deadlines can be met generally (soft real-time) or deterministically (hard real-time). It is valued more for how quickly or predictably it can respond to a particular event than for the amount of work it can perform over a given period of time. Such applications in embedded systems include data processing, automotive industry, healthcare, household appliance controllers, medical monitoring equipment, and consumer electronics.

Every care has been taken to eliminate misprints, omissions and errors but it is too much to expect that no inaccuary/misprint/error has crept in and the author would be grateful to the readers for bringing to his notice any such errors/misprints/omissions they may come across while going through the book.

The author will like to thank the readers (both teachers and students) who sent their valuable suggestions that became the basis for the revision of the book.

The author shall feel satisfied if the book meets the needs of the students for whom it is meant. Suggestions and criticism for improvement of the book are always welcomed.

The author hopes that book will fulfill the need of interested readers and welcome any suggestions towards the improvement of the book.

—Author

ACKNOWLEDGEMENTS

I would like to thank **Prof. Dr N. Satyanarayana, Principal, NITS, Hyderabad** for his valuable supervision and support during the preparation of this book.

I must express my deep appreciation to my wife **Mrs. Geetha Reddy** and my family for their patience, co-operation and understanding which made the tedious task of writing this book much easy to endure.

Special thanks to **N. Uday Kumar** (Research Scholar, ECE, SVU-Tirupathi), **Mr G. Pradeep Reddy** (Assist. Professor, ECE), **Mr M. Ravi Reddy** (Assist. Professor, EEE), **Mr Venkateswar Reddy** (Assist. Professor, EEE) **Mr N. Srinivas Reddy (MS), Mr G. Ram Mohan (MS),** for constant encouragement.

I would like to express my sincere thanks to **Dr K. Lakshma Reddy** chairman of **KLR Group of Institutions. Palvoncha** for constant inspiration and encouragement.

Great care has been taken to produce this book free from errors, but some may remain.

I will therefore accept the pleasure any information, which will draw the attention to such mistakes and errors. Criticism about the book shall always be appreciated.

—Author

CONTENTS

Unit 1

INTRODUCTION

- Introduction to UNIX
- Overview of commands
- File I/O (open, create, close, lseek, read, write)
- Process Control (fork, vfork, exit, wait, waitpid, exec)
- Signals
- Interprocess Communication (pipes, fifos, message queues, semaphores, shared memory).

The UNIX 0.5 started on a cast-off DEC PDP-7 at Bell Laboratories in 1969. Ken Thompson with ideas and support from Rudd Canaday, Aoug Melbroy, Jol Ossanna and Dennis Ritchie, wrote a small general purpose time-sharing system comfortable enough to attract enthusiastic users and eventually enough credibility for the purchase of a larger machine—a PDP-11/20. One of the early users was Ritchie, who helped move the system to the PDP-11 in 1970. Ritchie also designed and wrote a compiler for the C programming language in 1973. Ritchie and Thompson rewrote the UNIX kernel in C, breaking from the tradition that systen software is written in assembly language.

What makes the UNIX sys. so successful? We consider several reasons. First, because it is written in C, it is portable—UNIX sys. run on a range of computers from UPS to the largest main-frames this is a strong commercial advantage.

Second, the source code is available and written in a Hi-level language which makes the sys. easy to adapt to particular requirements. Finally, most important it is a good operating system, specially for programmers. The UNIX programming environment is usually rich and productive.

WHAT IS "UNIX"?

In the narrowest sense, it is a time-sharing O.S. kernel a program that controls the resources of a computer and allocates them among its users. It lets users run their programs; it controls the peripheral devices (discs, terminals, pointer connected to the machine and it provides a file

system, that manages the long-term storage of information such as programs, data and documents. In a broader sense, "UNIX" is often taken to include not only the kernel, but also essential programs like compilers, editors command languages, programs for copying and printing files, and so on.

UNIX is an operating system which was first developed in the 1960s, and has been under constant development over since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available, for example, in a telnet session.

TYPES OF UNIX

There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/ Linux, and MacOS X.

Here in the School, we use Solaris on our servers and workstations, and Fedora Linux on the servers and desktop PCs.

THE UNIX OPERATING SYSTEM

The UNIX operating system is made up of **three** parts;

- Kernel
- Shell
- Programs.

The Kernel

The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the file store and communications in response to system calls.

The Shell

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt (% on our systems).

FILES AND PROCESSES

Everything in UNIX is either a file or a process.

A process is an executing program identified by a unique PID (process identifier).

A file is a collection of data. They are created by users using text editors, running compilers etc.

Examples of files:

- a document (report, essay etc.);
- the text of a program written in some high-level programming languages;
- instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
- a directory, containing information about its contents, which may be a mixture of other directories (sub-directories) and ordinary files.

THE DIRECTORY STRUCTURE

All the files are grouped together in the directory structure. The file-system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called **root** (written as a slash /)

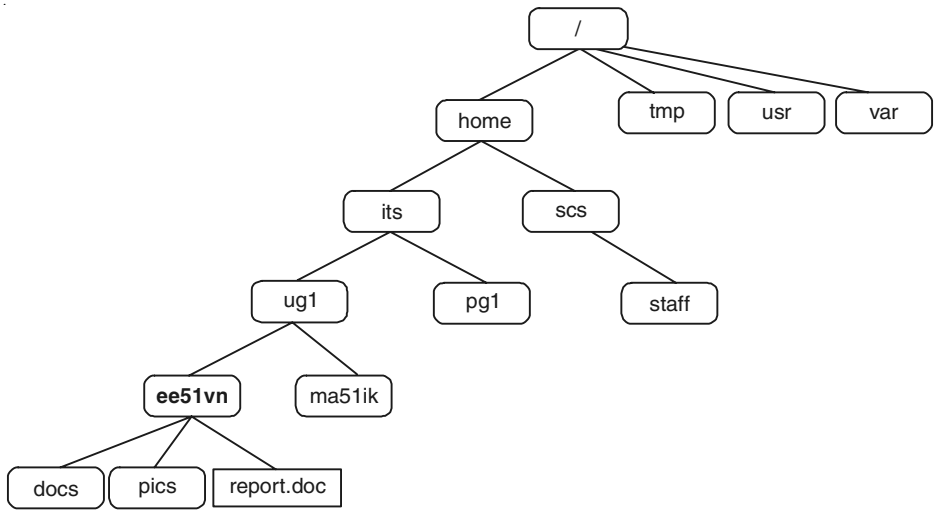


Fig. 1.1

In the diagram above, we see that the home directory of the undergraduate students **"ee51vn"** contains two sub-directories (**docs** and **pics**) and a file called **report.doc.**

The full path to the file **report.doc** is **"/home/its/ug1/ee51vn/report.doc".**

STARTING AN UNIX TERMINAL

To open an UNIX terminal window, click on the "Terminal" icon from the drop-down menus.

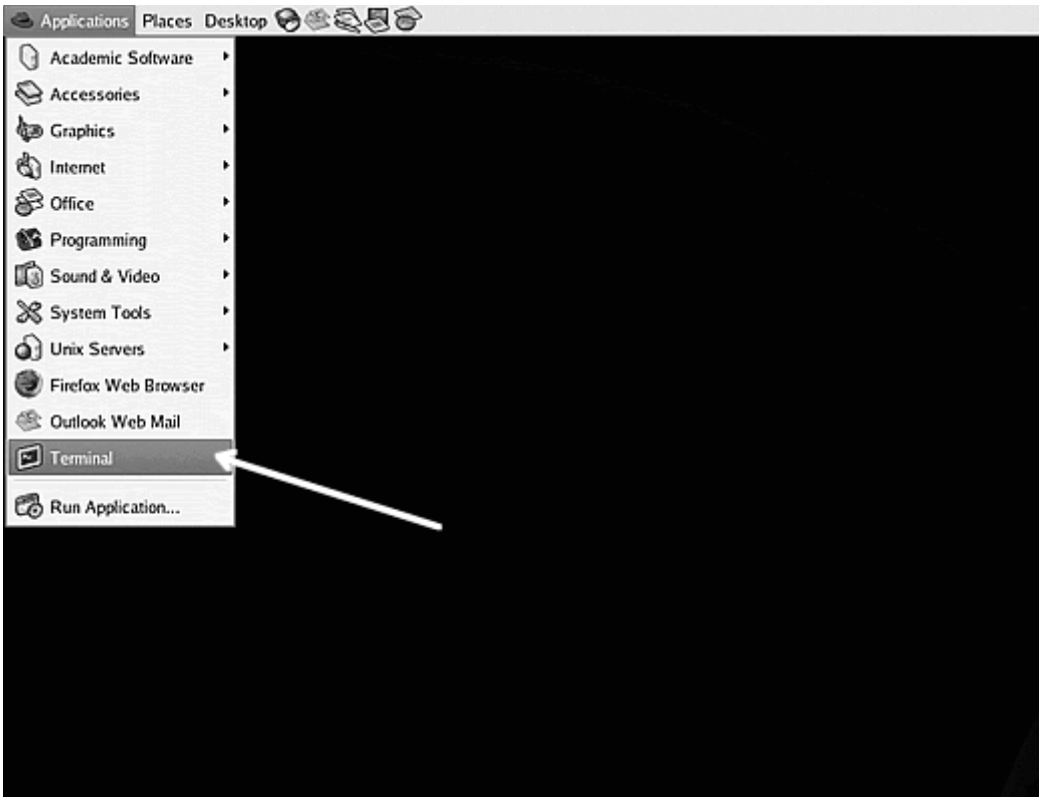


Fig. 1.2

An UNIX Terminal window will then appear with a % prompt, waiting for you to start entering commands.

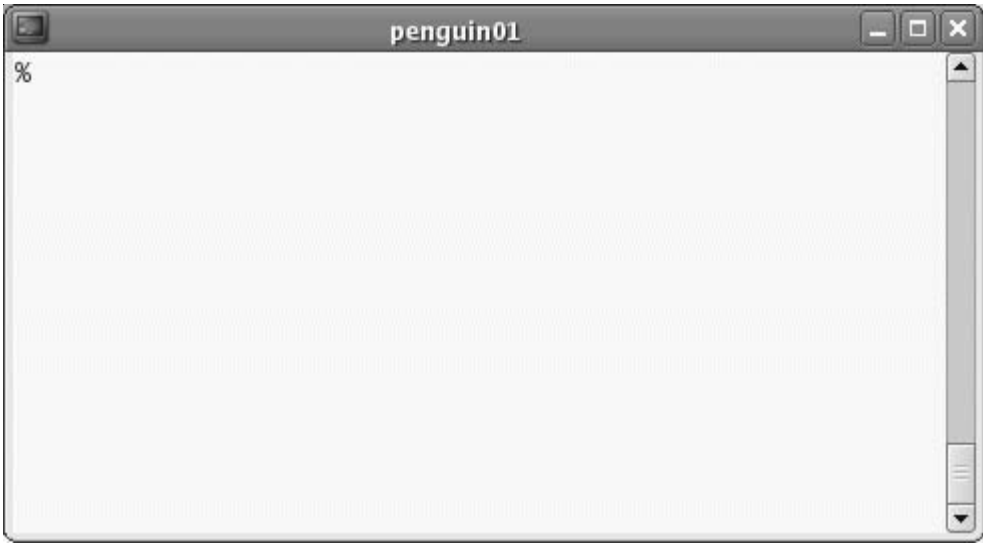


Fig. 1.3

OVERVIEW OF COMMANDS IN UNIX

- **tail**

  These commands used to print last 10 lines in a file.

  syntax: $ *tail* file name

  $ *tail*-1 file name print only last line of file

  $ *tail*-n file name print last n lines of file

  $ *tail*+n file name start printing file at line n

- **compare**

  These are used to print location of first difference.

  syntax: $ cmp file1 file2

- **difference**

  These are used to print all differences b/w files.

  syntax: $ diff file1 file2

- **pwd (print working directory)**

  $ pwd /usr/you

  This is says that you are currently in which directory, in directory user, which in turn is in the root directory. Which is conventionally called just '/'. The / characters separate the components of the name: The limit of 14 characters mentioned above applies to each component of such a name.

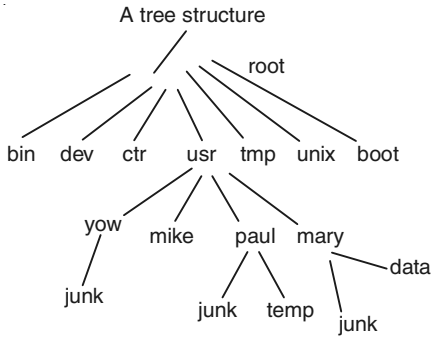


Fig. 1.4

→ Cat is used to display the content of file on the screen.

syntax: $ cat / usr / you / junk

- **ls**

  list names of all files in current directory.

  syntax: $ls ↵

$ls file names ↵	list only the named files
$ls-t	list in time order, most recent first
$ls-l	list long: more information; $lsuslt

$ls-u list by time last used; $lsuslu, $lsuslut

$ls-r list in reverse order; $ -rt, -rlt

- **edit**

  syntax: $ed file name edit named file

  Ex.: $ed /usr/mary/data

- **copy**

  syntax: $cp file1 file2 copy file1 to file2, overwrite file2 if it exists

  Ex.: $cp /usr/mary/data data

  make your own copy of one of her files

- **move**

  syntax: $ mv file1 file2 move file1 to file2

- **changing directory**

  This is used to changing your current directory.

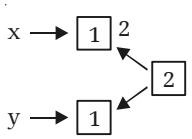
  syntax: $cd /usr/mary

- **make a directory (create a directory)**

  This is used to make a new directory

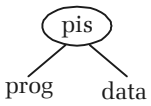
  syntax: $mkdir book $mkdir path x y

  $mkdir pis/prog pis/data



inode – index node

link – to link files



$ ln × Z increment by one x, y

- **remove directory**

  To remove the directory, remove all the files in it then cd to the parent directory or book and type

  syntax: $ rmdir book

  $ rmdir pis rmdir will only remove an empty directory

- **link**

  Used to link b/w files.

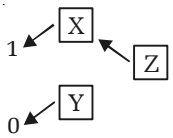
  syntax: $ ln cmp.ls_t cmp

  $ ln cmp.ls_t cmp.data

- **unlink (remove link)**

  syntax: $ unlink X decrement by 1

  $unlink Y



New Real-Time Operating Systems For Embedded Systems By K. Srinivasa Reddy

Publisher : Laxmi Publications ISBN : 9789381159736 Author : K. Srinivasa Reddy