

Kirkland & Ellis LLP Library



5 0628 01266903 9

UNIX[®] System V Network Programming

Stephen A. Rago



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

The programs and applications presented in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs or applications.

UNIX is a registered trademark of UNIX System Labs, Inc.

The publisher offers discounts on this book when ordered in quantity for special sales. For more information please contact:

Corporate & Professional Publishing Group
Addison-Wesley Publishing Company
One Jacob Way
Reading, Massachusetts 01867

Library of Congress Cataloging-in-Publication Data

Rago, Stephen A.

UNIX System V network programming/Stephen A. Rago.

p. cm. (Addison-Wesley professional computing series)

Includes index.

ISBN 0-201-56318-5 (hard)

1. Operating systems (Computers). 2. Unix System V (Computer file). 3. Computer networks. I. Title. II. Series.

QA76.76.063R34 1993

005.7'11—dc20

92-45276

CIP

Copyright © 1993 by Addison Wesley Longman, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

Text printed on recycled and acid-free paper.

ISBN 0-201-56318-5

9 101121314 MA 02 01 00 99

9th Printing March 1999

UNIX Programming

This chapter will give a brief overview of the UNIX programming environment. Although examples from this chapter are used throughout the rest of the book, readers familiar with this topic can skip to the next chapter without loss of continuity. The material in this chapter is intended to make the examples in the remainder of the book easier to understand. A more thorough treatment of the subject can be found in many of the books listed at the end of the chapter.

2.1 OVERVIEW

Figure 2.1 shows the architecture of the UNIX system. At the center is the *kernel*, the operating system core. It controls the computer's resources. *Drivers* are the parts of the operating system that control peripheral devices, such as terminals, printers, and disk drives. Although, strictly speaking, drivers are not part of the kernel proper, they are closely linked to the kernel and are considered part of the operating system. Drivers use the facilities provided by the kernel. In turn, the kernel relies on some drivers to provide vital services, such as access to devices used during system start-up. The terms "kernel" and "operating system" are often used interchangeably in the literature.

The operating system provides services to user applications through the *system call* interface. Each system call implements a basic service, for example input or output. Descriptions of each system call can be found in Section 2 of the *Programmer's Reference Manual*.

More elaborate functions are available in *library routines*, found in Section 3. These provide either enhanced versions of system calls, or facilities not found in system calls. An example of the former is the standard input/output (I/O) routines, which provide, among other things, buffering of data during input and output. An example of the latter is mathematical functions like `sqrt (3M)` and `sin (3M)`.

There are over 120 system calls in System V Release 4 (SVR4). There are even more library routines. We will not cover all the interfaces here. Instead, we shall

database. The first field is a network identifier that must uniquely identify the network and protocol on the system. The second field, called the network “semantics,” defines the type of service provided. Legal values currently include `tpi_clts` for connectionless transport service, `tpi_cots` for connection-oriented transport service, `tpi_cots_ord` for connection-oriented transport service with orderly release, and `tpi_raw` for pseudo-transport interfaces to other protocols, such as transport interfaces to internetworking protocols. In SVR4, the `tpi_raw` interface is used to access ICMP and IP.

The third field contains flags, or if no flags are specified, a dash (-). The only flag defined so far is `v`, indicating the network entry is visible to the NETPATH library routines (described shortly). An administrator can temporarily stop applications from finding a network by replacing this flag with a dash.

The fourth field contains a name for the protocol family. The Internet protocols are grouped under the name `inet`, for example. A dash can be used if no family name exists. Examples of different protocol families can be found in `<sys/netconfig.h>`. The fifth field specifies the name of the protocol supported by the network device. Again, a dash can be used to indicate that no protocol name exists.

The sixth field contains the pathname of the device to use when accessing the network and protocol. For a transport protocol, this is the name the application would pass to `t_open(3N)`. The seventh field is a comma-separated list of dynamic shared libraries that contain the network protocol’s name-to-address translation routines. A dash can be used in this field to indicate that no library routines exist.

By requiring that the library routines be implemented as dynamic shared libraries, the routines can be shared by multiple programs and will not be linked with a program until execution time. The deferred linking is made possible through the use of position-independent code (PIC) in the library routines. See `cc(1)` and `ld(1)` for the flags needed to generate PIC and dynamic shared libraries.

Two sets of library routines can be used to access the information in `/etc/netconfig`. The network-configuration library routines access the information line by line, and the NETPATH library routines access the information based on a shell variable. The network-configuration library routines and NETPATH library routines return information from the `netconfig` file through the `netconfig` structure:

```
struct netconfig {
    char      *nc_netid;      /* network identifier */
    ulong_t   nc_semantics;   /* service type */
    ulong_t   nc_flag;        /* flags */
    char      *nc_protobufmly; /* protocol family */
    char      *nc_proto;      /* protocol */
    char      *nc_device;     /* device pathname */
    ulong_t   nc_nlookups;    /* number of libraries */
    char      **nc_lookups;   /* list of library names */
};
```

Each member of the `netconfig` structure corresponds to a field in