

*Computer Systems  
Series*

---

# ***The LOCUS Distributed System Architecture***

---

*Gerald J. Popek and  
Bruce J. Walker*

*The MIT Press*

# **The LOCUS Distributed System Architecture**

**edited by**

**Gerald Popek and Bruce J. Walker**

**The MIT Press**  
**Cambridge, Massachusetts**  
**London, England**

**Copyright © 1985 by The Massachusetts Institute of Technology**

**All rights reserved. No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.**

**This book was printed and bound in the United States of America.**

**Publisher's note:** This format is intended to reduce the cost of publishing certain works in book form and to shorten the gap between editorial preparation and final publication. Detailed editing and composition have been avoided by photographing the text of this book directly from the author's typescript or word-processor output.

**Library of Congress Cataloging-in-Publication Data**

**Main entry under title:**

**The LOCUS distributed system architecture.**

**(MIT Press series in computer systems)**

**Bibliography: p.**

**Includes index.**

**1. LOCUS (Computer operating system) 2. Computer architecture. 3. Electronic data processing—Distributed processing. I. Popek, Gerald. II. Walker, Bruce J.**

**III. Series.**

**QA76.76.063L63 1985 005.4'3 85-18157**

**ISBN 0-262-16102-8**

## **Contents**

<b>Figures</b>	<b>ix</b>
<b>Series Foreword</b>	<b>xi</b>
<b>Preface</b>	<b>xiii</b>
<b>Acknowledgments</b>	<b>xvii</b>
<b>1      The LOCUS Architecture</b>	<b>1</b>
1.1    Architectural Overview and Document Roadmap	1
1.2    Transparent, Distributed Operation	2
1.3    High Reliability	2
1.4    High Performance	3
1.5    Unix Compatibility	3
1.6    Heterogeneous Machines and Systems	4
1.7    Guide to This Document	4
<b>2      Distributed Operation and Transparency</b>	<b>6</b>
<i>Gerald J. Popek</i>	
2.1    The Concept of Transparency	7
2.2    Dimensions of Transparency	9
2.3    Transparency and System Levels	10
2.4    Optimization Control	12
2.5    Naming	14
2.6    Heterogeneity	18
2.7    Error Modes	21
2.8    Local Autonomy and Transparency	23
2.9    Integration of Separate Networks	25
2.10   Summary	28
<b>3      The LOCUS Distributed Filesystem</b>	<b>29</b>
<i>Bruce J. Walker and Stephen H. Kiser</i>	
3.1    Filesystem Overview	29
3.2    Remote Service Control Flow	30
3.3    Static Picture of the Filesystem	33
3.4    Accessing the Filesystem	40
3.5    Synchronizing Distributed Filesystem Activity	48

3.6	Filesystem Support for Interprocess Communication	57
3.7	File Replication and Partitioned Operation	62
3.8	High Performance	66
3.9	Gateways	68
3.10	Other Issues	69
3.11	Summary	72
<b>4</b>	<b>Remote Tasking</b>	<b>73</b>
<i>David A. Butterfield and Richard M. Matthews</i>		
4.1	User View	73
4.2	Process Naming	75
4.3	Network Processes	76
4.4	Swapping and Paging	83
4.5	Process Tracking	84
4.6	Signals	89
4.7	Summary	89
<b>5</b>	<b>Dynamic Reconfiguration of LOCUS</b>	<b>90</b>
<i>Robert M. English</i>		
5.1	Requirements for the Reconfiguration Protocols	91
5.2	Protocol Structure	92
5.3	The Partition Protocol	93
5.4	The Merge Protocol	94
5.5	The Cleanup Procedure	95
5.6	Protocol Synchronization	95
5.7	Summary	97
<b>6</b>	<b>Heterogeneity</b>	<b>98</b>
<i>Gerald J. Popek</i>		
6.1	Machine-Independent Architecture and Implementation	98
6.2	Heterogeneous Hardware in a LOCUS Network	99
6.3	The LOCUS Kernel in a Heterogeneous CPU Network	99
6.4	Hidden Directories	102
6.5	Automatic Execution Site Selection	104
6.6	Summary	105

<b>7</b>	<b>System Configuration, Installation and Administration</b>	<b>106</b>
<i>Greg I. Thiel</i>		
7.1	System Configuration	106
7.2	LOCUS Installation	109
7.3	System Administration	116
7.4	Summary	118
<b>8</b>	<b>Conclusions</b>	<b>120</b>
8.1	Experiences	120
8.2	Unix Retrospective	122
8.3	Architectural Lessons	123
8.4	Futures	124
<b>Appendices</b>		<b>125</b>
Appendix A—Additional LOCUS System Calls		125
Appendix B—LOCUS Internal Network Messages		127
<b>Bibliography</b>		<b>140</b>
<b>Index</b>		<b>147</b>

## Preface

The distributed computing system described here was developed at the University of California at Los Angeles (UCLA) during the late 1970s and early 1980s. It was originally called LOCUS (Local Computer User System), and was later renamed Locus. It was developed by a small group of researchers and students under the direction of Dr. David H. Bailey.

Distributed, interconnected and cooperating computing systems are an increasing reality in many environments, and both technological and social forces promise to accelerate this trend. VLSI and other simplifications are making desktop computers powerful engines and collections of these can have an aggregate compute power equivalent to a very large machine, but at far less cost. Therefore, in those cases where any single computation does not require the very high speed of the large machine, the small ones can be far more economical. It is also much easier in most organizations to justify low cost purchases; thus the result is often a rapid growth of investment in small computers, even without a rational evaluation of the economic advantage. The many advantages of desktop workstations, especially when coupled with larger mainframe machines for work which overflows the desk, are well known.

Unfortunately, it generally becomes necessary to make the resulting collection of computers work together smoothly, a daunting task not present in the single large machine case. This is the distributed computing problem. The LOCUS research effort, begun at UCLA in the late 1970s, attempted to understand how application software and computer networks should interact. The goal was to allow full network access to users who were "mere mortals", without special training. At the time, developing network software was so difficult that most people avoided it. The fundamental concept of *network transparency* grew out of these early efforts, and it was decided to try to build an actual transparent distributed operating system, both to demonstrate the feasibility of this concept, and also to provide a means to further understand its ramifications. By early 1981, a prototype system on a small collection of PDP-11s connected by an early ring network was operational. However, it wasn't until a year and a half later that enough of the associated technical problems and issues were understood and solved well enough that the system, then operational on VAXes, could be made generally available to users other than its original developers.

While many of the early users were enthusiastic in principle, they were far less happy with the non-product quality of the system. It was clear that the Locus approach had great utility but it was beyond the scope of a university research group to provide a finished production product.

Consequently, efforts were made to find a commercial vehicle to produce a product quality derivative of the concepts demonstrated by the research. Early in 1983, that work began at Locus Computing Corporation, an effort which would eventually encompass a number of different machine types, new concepts, largely reimplemented software, and a commitment of manpower and intensity which would dwarf that done earlier at UCLA.

There are several lessons worth learning from this experience. First, distributed functionality should be largely hidden from application programs and users. Providing a convenient distributed computing environment, as represented by a very high degree of transparency, is difficult. But, building software on top of such a base is essentially as easy as building for a single machine. Consequently, it is clear that these facilities ought to be provided in such a manner that maximum use of them may be made. An application should not be required to deal with knowing whether a resource it wishes to interact with is local or not, or knowing specialized means of interacting with remote resources, any more than high level language programmers should be required to write I/O channel programs, or overlays to provide the equivalent of virtual memory. The unnecessary cost and pain is inexcusable. The operating system is thus an appropriate vehicle for support of transparency.

Second, it is exceedingly difficult to reduce the elapsed time between the inception of a research idea and its commercial availability below the typical decade so often quoted. LOCUS has done better, but due only to a combination of vision within a major corporation, and an unusual level of dedication from its original conceivers and developers, many of whom participated in its entire history of maturation.

This book is intended to convey to the reader both the concepts which are key to the distributed system architecture, and "truth" in the form of an actual case study of what is required to support such functionality. The system as described in these pages has been fully operational for some time; ideas, however appealing on the surface, which were not successfully implemented in a product quality way, complete with error handling, integration with other features, and stability under a wide range of conditions are not included in the book.

In closing, it is difficult to convey how many things had to "go right", and how many people's enormously dedicated support was essential in order to develop these concepts and make a resulting system product publically available. The director and research managers at the U.S. Department of Defense research agency ARPA, staff at UCLA, and

the dedicated Locus Computing Corporation team are obvious. Numerous people within IBM and especially the IBM Palo Alto Science Center have invested considerable effort in "making it happen". Heartfelt thanks to all.

*Gerald J. Popek  
Santa Monica, California  
June 1985*

## **Appendix A**

### **Additional LOCUS System Calls**

Below is a list of the system calls added to a standard System V or 4.1 BSD Unix kernel to allow programs to optionally interrogate or control the added functionality in LOCUS. The first set of calls are largely available to application programs (some of the *set* calls are superuser only) while the second set are used for system administrator programs

chhidden	Change hidden attribute of directory, making hidden directories from regular directories and vice versa.
commit	File changes made up to the point of the commit call are made permanent.
dfstat	Get file status of open file, including new LOCUS fields.
dstat	Extended version of stat, giving new LOCUS fields.
getcommit, setcommit	Return or change the current default (commit/abort) action to be taken when an exception occurs.
gethpath, setpath	Return or change the current hidden directory search path.
getlocal, setlocal	Return or set name of the current process's <LOCAL> filesystem.
getsites, setsites	Return which sites are in your partition or set which sites may be in your partition. Setting is restricted to superuser.
getxperm, setxperm	Return or change list of sites on which the current process is permitted to execute or have children execute. Setting is restricted to superuser.
getxsites, setxsites	Return or change the execution site list used for future fork, exec, run, and migrate calls.
kill3	Similar to kill except an additional argument is passed. This is useful to signal SIGMIGRATE where a site needs to be specified.
lknod	Extended version of mknod, allowing the setting of the site a device resides on.
migrate	Move calling procedure to another site.
readlink	Return the filename to which a symbolic link points.

runl, runle, runv, runve	Create a new process and transfer control to it.
sabort	File changes made up to the point of the abort call after the last commit are thrown away.
select	Return information on which file descriptors are ready for I/O. This allows synchronous I/O multiplexing.
setxuid	Use real user id or group id for subsequent invocations of run.
site	Return the site number at which the currently running or specified process is executing.
symlink	Create a symbolic link (similar to 4.2 feature).

Below is the second list, which are system calls used by system administrator functions like filesystem reconciliation.

netstart	Initiate network activity.
probe	Validate status of a site by attempting to communicate with it.
raccept	Give me a function to perform - used to allow the kernel to have processes started up (eg. filesystem reconciliation)
propv	Tell the kernel to propagate a given version of a given file. Superuser only and used by filesystem reconciliation process.
chlwm	Change or interrogate the low water mark of a filegroup pack. Superuser only and used by filesystem reconciliation process.

The MIT Press

Massachusetts Institute of Technology  
Cambridge, Massachusetts 02142

### The LOCUS Distributed System

#### Architecture

by Gerald J. Popek and Bruce J. Walker

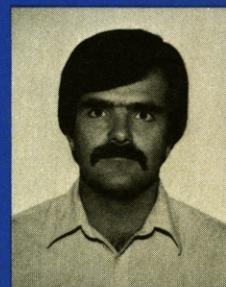
Computer systems consisting of many machines will be the norm within a few years. However, making a collection of machines appear as a single, coherent system — in which the location of files, servers, programs, or users is invisible to users who do not wish to know — is a very difficult problem.

LOCUS, a distinguished version of the popular operating system Unix, provides an excellent solution. It makes a collection of computers, whether they are workstations or mainframes, as easy to use as a single computer by providing a set of supports for the underlying network that is virtually invisible to users and applications programs. This "network transparency" dramatically reduces the cost of developing and maintaining software, and considerably improves the user model of the system. It also permits a variety of system configurations, including diskless workstations, full duplex I/O to large mainframes, transparently shared peripherals, and incremental growth from one workstation to a large network including mainframes with no effect on applications software required to take advantage of the altered configurations.

In addition to transparent, distributed operation, LOCUS features also include high performance and reliability; compati-



Gerald J. Popek



Bruce J. Walker

---

bility with Unix and support for heterogeneous machines and systems as well as database management; and architectural extensions to support bitmap graphics, extensive interprocess communication, internetworking, process snapshots for very high reliability and availability, and a B-tree package for data management.

Gerald Popek is Associate Professor of Computer Science at UCLA and President of Locus Computing Corporation in Santa Monica. *The LOCUS Distributed System Architecture* is included in the Computer Systems series, edited by Herb Schwetman.

POPLH

0-262-16102-8