ISSN 1512-1232

RESEARCH PURPOSE OPERATING SYSTEMS – A WIDE SURVEY

Pinaki Chakraborty

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi – 110067, India.
E-mail: pinaki_chakraborty_163@yahoo.com

Abstract

Operating systems constitute a class of vital software. A plethora of operating systems, of different types and developed by different manufacturers over the years, are available now. This paper concentrates on research purpose operating systems because many of them have high technological significance and they have been vividly documented in the research literature. Thirty-four academic and research purpose operating systems have been briefly reviewed in this paper. It was observed that the microkernel based architecture is being used widely to design research purpose operating systems. It was also noticed that object oriented operating systems are emerging as a promising option. Hence, the paper concludes by suggesting a study of the scope of microkernel based object oriented operating systems.

Keywords: *Operating system, research purpose operating system, object oriented operating system, microkernel*

1. Introduction

An operating system is a software that manages all the resources of a computer, both hardware and software, and provides an environment in which a user can execute programs in a convenient and efficient manner [1]. However, the principles and concepts used in the operating systems were not standardized in a day. In fact, operating systems have been evolving through the years [2].

There were no operating systems in the early computers. In those systems, every program required full hardware specification to execute correctly and perform each trivial task, and its own drivers for peripheral devices like card readers and line printers. The growing complexity of the computer hardware and the application programs eventually made operating systems a necessity. Initially, operating systems were not fully automatic as Hansen [3] defined an operating system as a set of manual and automatic procedures that enable a group of people to share a computer installation efficiently. It is fortunate enough for today's computer users that modern operating systems are fully automatic.

Historically, operating systems have been closely tied to the architecture of the computers on which they run [2,4]. Consequently, developments in computer hardware immensely affected the course of evolution of operating systems. In the rapid and somewhat chaotic growth of operating systems, many important concepts have emerged, disappeared, and then reappeared often in different forms [5]. Over the years, sustained research in operating systems gave rise to many novel concepts and ideas. Operating systems exist even today because they offer a reasonable way to solve the problem of creating a usable computing system [1]. Moreover, sophisticated operating systems increase the efficiency and consequently decrease the cost of using a computer [5]. A large number of operating systems of various types are available today for both research and commercial purposes, and these operating systems vary greatly in their structures and functionalities [1].

It is difficult to present a complete as well as deep account of operating systems developed till date. Early attempts to do so were made by Rosen [6] and Rosin [7]. But it will be too ambitious to make such an attempt now. So, this paper tries to overview only a subset of the available operating systems. Operating systems are being developed by a large number of academic and commercial organizations for the last several decades. Many of these projects were short lived and in several cases without much impact. Descriptive research literatures for many of these operating systems are

ISSN 1512-1232

not available. Information about such operating systems is often found scattered over the Web. The article on operating systems in the Wikipedia [8] and the pages hyperlinked to it, however, serves as a rich source of information in this field. This paper is highly indebted to the Wikipedia for miscellaneous information. Moreover, the technical details of the commercial operating systems are seldom discussed openly. Many commercial operating systems do not have any technological consequence and merely use concepts introduced in earlier research purpose operating systems. Another awkward issue is the availability of numerous versions of the commercial operating systems that in many cases do not have any significant enhancement. On the other hand, sometimes two absolutely different types of operating systems are marketed as two versions under the same name just for business reasons. As a result, commercial operating systems are of limited interest to the researchers working on the design and implementation of operating systems. This paper, therefore, concentrates on the research purpose operating systems with special emphasis to those that had deep impact on the evolution process. The aim of this paper is to provide a brief timely commentary on the important research purpose operating systems available today. Thorough analyses of all these operating systems are surely out of the scope of the paper.

Researchers like Herder [9], Tanenbaum and Woodhull [2], and Silberschatz *et al.* [1] have been using various criteria to classify the operating systems. In this paper, the research purpose operating systems are classified into three categories using an ideological criterion. The three categories are Unix-like operating systems, non-Unix-like operating systems and object oriented operating systems. The first two categories are mutually exclusive (Figure 1). However, an object oriented operating system can be either a Unix-like operating system or a non-Unix-like operating system. The operating systems belonging to these three categories are discussed in the next three sections.

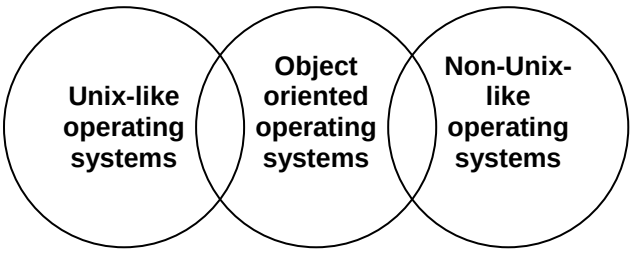

Figure 1: Three categories of research purpose operating systems.

2. Unix-Like Operating Systems

An operating system is said to be a Unix-like operating system if it behaves similar to the Unix operating system [8]. The Unix-like operating systems are typically implemented in high level languages, predominantly C, which is an achievement in itself. These operating systems are widely used for research and they have been able to standardize several new techniques including the handling of devices as files. These operating systems have been ported to almost all types of computer architectures. They are highly portable and generally multiuser in nature. The Unix-like operating systems are extremely interoperable and the application programs developed for one of them often work acceptably for the others. Apart from their importance in research, quite a few of the Unix-like operating systems have seen substantial commercial success. A taxonomy of the Unix-like operating systems has been developed by Eric S. Raymond who classified them into three subcategories as listed below.

1. *Genetic Unix Operating Systems*. These are the operating systems that have some historical connection with the original Unix operating system and share some part of the source code of the original Unix operating system.
2. *Branded Unix Operating Systems*. These are the operating systems that have been determined by the Open Group to meet the Single UNIX Specification.

ISSN 1512-1232

3. *Functional Unix Operating Systems*. These are the operating systems that behave roughly like the Unix operating system.

Numerous Unix-like operating systems have been developed for research and academic purposes. Some of the important ones, starting with Unix itself, are studied next.

2.1. Unix

Unix [10–13] is a monolithic kernel based operating system. It was developed by team of researchers including Dennis Ritchie and Ken Thompson at the Bell Labs. It was implemented in C and was one of the first operating systems to be implemented in a high level language. Over the years, Unix has become highly influential and perhaps the most widely studied research oriented operating system. Additionally, it has also seen certain degree of commercial success. Now, Unix based operating systems have been split into various branches and they are developed by various commercial vendors and nonprofit organizations. Unix has been ported to almost all types of platforms. It runs successfully on a wide range of computer systems including personal computers, servers and supercomputers. From the very beginning, Unix was designed to be a highly portable, multitasking and multiuser operating system. Unix introduced several new concepts like storing of data using plain text, hierarchical file systems and treating devices as files. The Unix operating system consists of a monolithic kernel and number of utility programs. One of the most important utility programs is the command interpreter or the shell. Unix implements the command interpreter as an ordinary user level program. Additional commands are provided as separate programs. New commands can be added by writing such separate programs. Since the command interpreter is an ordinary user level program, it can be replaced by some customized command interpreters. This option gave rise to a large number of Unix shells. The Unix shells use the same language for interactive commands and for scripting. Consequently, shell programming in Unix has become vastly popular.

2.2. MINIX

MINIX is a widely studied pedagogical operating system. It is being developed by Andrew S. Tanenbaum and his fellow researchers at the Vrije Universiteit in Amsterdam. MINIX is a free open source software. The name, MINIX, derives from the words ‘minimal’ and ‘Unix’. MINIX is one of the first microkernel architecture based Unix-like operating systems. It is also one of the first Unix-like operating systems that do not share the source code of Unix. MINIX has also inspired the creation of the Linux kernel. Several major and minor versions of MINIX are now available. The latest version, MINIX 3 [2,14], is a pure microkernel based operating system. However, its predecessors were hybrid microkernel based operating system [15]. MINIX has been implemented predominantly in C with a little use of assembly language. MINIX has been ported and runs on various platforms including Intel x86, Motorola 68000 and SPARC.

2.3. Linux

Linux [16] was developed by Linus Torvalds as a noncommercial replacement for the MINIX operating system at the University of Helsinki. Alike MINIX, Linux is a functional Unix-like operating system. However, unlike MINIX, Linux used a monolithic kernel. The Linux operating system gradually became a commercial system and today it is quite successful as a commercial operating system. The Linux kernel was first released for the Intel x86 platform. The kernel was augmented with system utilities and libraries from the GNU project to create a usable operating system, which later led to the alternate term GNU/Linux. Linux is now available from different Linux distributions and the packages contain the Linux kernel, or a modified version of it, along with a variety of other software tools. Linux is well known for its performance in servers and is well accepted in the industry. Linux has been ported to various platforms and today it is used as an operating system for a wide variety of computer hardware including desktop computers, supercomputers and embedded devices such as mobile phones and routers.

ISSN 1512-1232

2.4. Solaris

Solaris is a Unix-like operating system [17,18]. It is being developed by the Sun Microsystems. Solaris is highly successful in symmetric multiprocessor systems where it can support a large number of processors. Many versions of Solaris are available. These versions illustrate a range of concepts and ideas that has been used in the Solaris project over the years. All versions of Solaris use monolithic kernels. Solaris is used in various types of computers ranging from personal computers and workstations to distributed systems and supercomputers. Solaris has been ported to a large number of platforms including Intel x86, SPARC and PowerPC.

2.5. XINU

XINU is a recursive abbreviation for XINU Is Not Unix. It is a Unix-like operating system [19–21]. It is being developed by Douglas E. Comer at the Purdue University. XINU is used as a pedagogical tool for operating system and networking courses in many universities around the globe [8]. It has also been deployed in some commercial establishments. XINU has been ported to several platforms including Intel x86, DEC LSI-11, Sun-2 and Sun-3 workstations, PowerPC G3 and MIPS.

2.6. Plan 9

Plan 9 is a distributed operating system [22,23]. It was developed as the research successor to Unix by the Computing Sciences Research Center at the Bell Labs. As a distributed operating system, Plan 9 has made major breakthroughs in implementing system wide transparency. Plan 9 provides users a terminal independent working environment. This achievement can be largely attributed to the use of the 9P protocols to access both local and global resources. Plan 9 uses Unicode as the native encoding throughout the system after Ken Thompson invented UTF-8.

2.7. Inferno

Inferno is an enhancement over the old Plan 9 operating system [24]. It was developed by the Vita Nuova company. Inferno was developed primarily to support efficient creation and execution of distributed applications. Inferno can be run as a guest operating system on various operating systems including Windows and Linux. Alternatively, Inferno can be executed as the native operating system on several platforms including Intel x86, SPARC and PowerPC. In each configuration, Inferno presents the same standard interfaces to the application software layer. A communication protocol called Styx is used to access both local and remote resources. Application programs are written in the Limbo programming language and are executed by a bytecode interpreter or just-in-time compiler called Dis. Inferno has many similarities with the Java Virtual Machine [25].

2.8. Plan B

Plan B is a distributed operating system [26]. It is implemented as a set of user programs running on the top of the Plan 9 system. Plan B is designed to work in dynamically reconfigurable distributed computing systems. In Plan B, same protocols are used to access the local and remote entities. This enhances transparency. As an attempt to create stateless server processes, application programs avoid establishing connections with resources, by using calls that accept file names instead of file descriptors.

2.9. IRIX

IRIX is a monolithic kernel based operating system [27]. It was developed by the Silicon Graphics Inc. IRIX runs on 32-bit and 64-bit MIPS architecture based workstations and servers. IRIX supports real time disk and graphics input/output. It was one of the first Unix-like operating

ISSN 1512-1232

systems to feature a graphical user interface for the main desktop environment. IRIX has been widely used in the computer animation industry and for scientific visualization [8].

2.10. LynxOS

LynxOS is an abbreviation for Lynx Operating System. It is being developed by the LynuxWorks company. LynxOS is a Unix-like real time operating system [28]. Historically, it was developed to run on a Motorola 68010 processor. However, it has been thoroughly modified and revised several times. Current versions can be ported on various platforms and feature POSIX conformance as well as Linux compatibility. LynxOS is typically used in real time embedded systems in various application domains including avionics, aerospace, military, industrial process control and telecommunications. LynxOS is a hard real time system and its components display absolute determinism. The LynxOS kernel uses a thread model that enables interrupt handler routines to be exceptionally small. As a result, predictable response times can be ensured.

3. Non-Unix-Like Operating Systems

There are several research purpose operating systems that have been designed and implemented from the ground up without using any major principle of the Unix operating system. These operating systems form the widely varied class of non-Unix-like operating systems. Non-Unix-like operating systems include several early operating systems like THE and MULTICS which have immense influence on both Unix-like and non-Unix-like operating systems developed since then. Since the non-Unix-like operating systems have been developed using various dissimilar design methodologies, they illustrate a diverse collection of principles that can be used to develop operating systems. The non-Unix-like operating systems are best known for their roles in developing the structured design of operating systems and in the evolution of the concept of microkernels. Some important non-Unix-like operating systems are studied in this section.

3.1. THE

The THE multiprogramming system was one of the early operating systems. It has influenced almost all coming generations of operating system. It was developed by Edsger Wybe Dijkstra and his fellow researchers at the Eindhoven University of Technology. The name is an abbreviation for Technische Hogeschool Eindhoven, the then name of the university. The THE operating system was developed for the Electrologica X8 platform and ALGOL was the only language supported by it [29]. The operating system was implemented in assembly language. The Electrologica X8 computer used 27 bit words and had a core memory of 32K. The computer used a drum type backing store. The inputs were provided through paper tape readers, and the outputs were obtained through paper tape punches and printers. The THE operating system was the first layer structured operating system. The operating system components were divided in six mutually exclusive and exhaustive layers. Each layer was allowed to access only the services provided by its lower layers. Layer 0 was the lowest layer. It was responsible CPU scheduling, interrupt handling and context switching. As a result, it was solely creditworthy for the multiprogramming aspects of the operating system. Layer 1 was responsible for allocating memory to the various processes. Layer 2 was responsible for the interprocess communication as well as communication between the operating system and the console. The Layer 2 also used semaphores for synchronization and the Banker's algorithm for deadlock avoidance. Layer 3 was responsible for handling all input/output devices and buffers for these devices. Layer 4 consisted of user mode programs for compilation, execution and printing of user programs. The user programs were scheduled on a priority basis. Layer 5 was responsible for the overall control of the system and was sometimes called the system operator. Soon after the success of the layered structure of THE, the MULTICS operating system was developed with a ring segmented kernel. This concept of division of the kernel into layers was used in many subsequent operating systems including Windows NT and Mac OS X. However, the ensuing operating systems typically had fewer layers.

ISSN 1512-1232

3.2. MULTICS

MULTICS is an abbreviation for Multiplexed Information and Computing Service. It was one of the important early operating systems and it has significantly influenced numerous operating systems developed in the subsequent decades. The development of MULTICS was started as a joint venture by the Massachusetts Institute of Technology, General Electric and Bell Labs. However, Bell Labs later dropped out of the project and Honeywell subsequently acquired General Electric's computer business. MULTICS was conceived to be a commercial product but it was never widely accepted in the industry. Nonetheless, the many innovative and novel ideas of MULTICS had a huge impact on the future operating systems. MULTICS was designed to provide high availability to a large number of users [30–35]. It aimed at developing a computing utility similar to telephone exchange or electricity grid. The software structure was kept highly modular. The system was scalable and extra resources including computing power, main memory and disk storage were easily augmentable. MULTICS was also notable for its early emphasis on computer security by design, especially with the use of hardware supported ring oriented security. MULTICS had a hierarchical file system and the filenames could be of almost arbitrary length and syntax. Symbolic links between directories were also supported. Security at the file system level was implemented using separate access control list for each file. This enabled flexible information sharing as well as complete privacy as per requirements. Mechanisms for performance analysis and adaptive performance optimization were also included. MULTICS was implemented in PL/I and it became one of the first operating systems to be implemented in a high level language [36,37].

3.3. V

V is a microkernel based operating system [38]. It was developed by the Distributed Systems Group at the Stanford University. The project was led by David R. Cheriton. The V operating system is sometimes considered as a successor of the Thoth and the Verax operating systems [8]. The V operating system is best known for its interprocess communication mechanism. It supports multithreading and interthread communication by synchronous message passing using fixed length messages. The same mechanism is also used to implement remote procedure calls.

3.4. Mach

Mach is an operating system microkernel [39]. It was developed at the Carnegie Mellon University. Richard F. Rashid and Avie Tevanian were two important developers in this project. Mach was developed to support advanced research in operating systems including the study of the role of operating systems in parallel and distributed systems. Mach is one of the earliest microkernels and it was the logical successor of the Accent kernel [40]. It was designed as a replacement of the Unix kernel. The Mach project tried to formalize the concept of interrelationships between tasks, threads, ports and messages.

3.5. L4

L4 is a family of microkernel based operating systems [41,42]. L4 was gradually enhanced to achieve better platform independence, security and robustness. There have been various re-implementations of the original design of L4. L4Ka::Hazelnut and L4Ka::Pistachio were developed at the University of Karlsruhe, L4/MIPS and L4/Alpha were developed at the University of New South Wales and Fiasco was developed at the Technische Universitat Dresden. Consequently, the name L4 has been generalized and it now applies to the entire family of microkernel based operating systems that include the L4 microkernel interface and its variants. The L4 microkernel was kept minimal in size and functionality. It provides only some basic mechanisms like a thread model and synchronous thread communication, scheduling, and an address space abstraction.

3.6. EROS

ISSN 1512-1232

EROS is an abbreviation for Extremely Reliable Operating System which is a pure capability based operating system [43–45]. It was developed jointly by the EROS Group, the Johns Hopkins University and the University of Pennsylvania. EROS is purely a research operating system and it was never used for any commercial purpose. EROS supports both data persistency and process persistency. Both data and processes are saved on the secondary storage when the system is switched off and are available whenever the system is restarted. The EROS operating system provides support for component based software development. In component based software development, a system is developed as a collection of autonomous components. The components are usually small and individually testable. This facilitates identification of flaws and bugs. The isolation of the components from each other limits the scope of the damage that may occur in the case of failure of a particular component. Consequently, a robust and secure system is obtained.

3.7. CapROS

CapROS is an abbreviation for Capability Based Reliable Operating System which is a microkernel based operating system [46]. It is an open source software. CapROS is being developed by the Strawberry Development Group and Charles Landau is the key developer. The CapROS operating system evolved from the EROS operating system. However, there was major a shift in the ideology. Unlike EROS, which was purely a research purpose operating system, CapROS was developed as a commercial operating system. CapROS is a pure capability based system and supports the principle of least authority. Like EROS, it supports both data and process persistencies. CapROS provides security and fault tolerance of very high quality. CapROS currently runs on processors developed by Intel and ARM.

3.8. Coyotos

Coyotos is a microkernel based operating system. It evolved from the EROS operating system. Coyotos is capability based and security focused in nature [47]. It is being developed at the Johns Hopkins University's Systems Research Laboratory. Jonathan Shapiro is the primary contributor in the development of the Coyotos operating system. Coyotos is being developed to be the first formally verified operating system. Accordingly, a new programming language, called BitC, and its complier, called BitCC, are also being developed simultaneously. The Coyotos operating system is being designed to be a pure capability based operating system and to support asynchronous communication.

3.9. Amoeba

Amoeba is a widely studied distributed operating system [48]. Amoeba is a microkernel based operating system which is often referred for its meticulous design. It was developed by Andrew S. Tanenbaum and his fellow researchers at the Vrije Universiteit in Amsterdam. From the very beginning, the Amoeba project emphasized on teaching and research. Accordingly, the Amoeba operating system is available as an open source software. As a distributed operating system, Amoeba is best known for its attempt to obtain transparency in a distributed computing system [49–51]. Amoeba tries to present a single system image of the distributed computer system to the users and tries its best to hide the intricacies of the distributed computing system. Although the Amoeba operating system was designed and implemented as an academic venture, today it is being used in industry and government establishments as well. Amoeba has been ported on several platforms including SPARC, Intel 386, Intel 486, Motorola 68030, Sun 3/50 and Sun 3/60.

3.10. House

House is an abbreviation for Haskell User's Operating System and Environment [52]. It is a purely research purpose operating system written in Haskell. It was developed to explore the scope of implementing system software in a functional programming language. And it proved quite successful in demonstrating the capability of high level programming languages in the field of

ISSN 1512-1232

system programming. The House operating system has a graphical user interface. Its network protocol stack provides basic support for Ethernet and various common protocols including TCP, UDP, IP, ICMP and TFTP.

3.11. ILIOS

ILIOS is an abbreviation for Interlink Internet Operating System [53]. It was developed by Rink Springer. ILIOS was developed primarily for routers and it focuses on various networking issues. However, it does no support multitasking and is largely interrupt driven. Nonetheless, its simple structure makes ILIOS a good pedagogical operating system.

3.12. Integrity

Integrity is a microkernel based real time operating system [54]. It is being developed by the Green Hills Software company. Integrity uses the velOSity microkernel. Integrity is developed for use in 32-bit and 64-bit embedded systems. It is highly reliable and fault tolerant. Integrity uses hardware level memory protection. It separates memory space for the operating system and application programs. As a result, application programs cannot interfere with the working of the operating system and with each other. Although Integrity is a non-Unix-like operating system, it is POSIX compliant. Integrity runs on several platforms including Intel x86, Blackfin, ColdFire, MIPS and PowerPC.

3.13. Nemesis

Nemesis is an operating system optimized for multimedia support [55]. It was developed jointly by the University of Cambridge, the University of Glasgow, the Swedish Institute of Computer Science and the Citrix Systems. It was observed in earlier operating systems that multimedia based application programs spend most of their time executing in the privileged kernel mode. This may lead to several pitfalls including possible breach in the security of the operating system. Nemesis was designed to overcome these problems. Consequently, Nemesis has an exceptionally small and lightweight kernel to support a minimal set of fundamental operating system functions. Rest of the operating system practicalities and the multimedia support are implemented as the user mode processes. Nemesis has been ported to several platforms including Intel x86, Alpha AXP and ARM.

3.14. Singularity

Singularity is a microkernel based operating system [56]. It is being developed by the Microsoft Corporation. Singularity is an attempt to develop a highly reliable operating system fully implemented using managed code [57]. The Singularity microkernel uses the new concept of software isolated processes. Singularity has been implemented using several languages. Most of the kernel and the device drivers are written in C#. The hardware abstraction layer is written in C++. The interrupt dispatch code is written in assembly language and C. Singularity uses some concepts used by the Inferno operating system.

4. Object Oriented Operating Systems

An operating system is said to be an object oriented operating system if and only if it internally uses an object model. Following an object model, an operating system can be designed and implemented as a set of types, each of which can be thought of as a kind of resource [58]. Some of these resources, like input/output devices, may have direct physical realizations. Alternatively, some resources may be abstract and without any realization at the computer hardware level. Semaphores, mailboxes and files are some common examples of abstract resources. Each resource, whether hardware realizable or not, is modeled as an object in an ideal object oriented operating

ISSN 1512-1232

system. Quite a few object oriented operating systems have been developed for the purpose of research and some of them are discussed here.

4.1. NeXTSTEP

NeXTSTEP is a hybrid kernel based object oriented operating system [59]. It was developed by NeXT. NeXTSTEP is a Unix-like operating system. It uses the Mach microkernel and some source code of BSD Unix. Though the runtime environment and the upper layers are object oriented in nature, the Mach microkernel is not object oriented. Hence, NeXTSTEP cannot be considered an object oriented operating system in the strictest terms. Although NeXT's endeavor was innovative and novel in many ways, NeXTSTEP was never commercially successful and it gained only small acceptance in the industry. However, NeXTSTEP later became the basis for Mac OS X. NeXTSTEP has been ported to various platforms including Intel x86, Motorola 68000, Sun SPARC and HP PA-RISC.

4.2. Athene

Athene is a suitable example of object based operating system [60]. It was developed by the Rocklyte Systems. The user environment of the Athene operating system consists purely of objects which are configured and linked dynamically. An object scripting language called Dynamic Markup Language is available to develop object based application programs that can be executed by the Athene operating system. Athene also provides support for sharing of objects. Objects instantiated in a shared memory can be shared by two or more processes with the help of proper mutual exclusion mechanisms. The objects in an Athene environment can be ported to Windows and Linux environments for the development of object oriented programs. This improves interoperability between the computer systems.

4.3. Choices

Choices is an object oriented operating system [61]. It was developed at the University of Illinois at Urbana-Champaign. It was implemented primarily in C++. The Choices operating system models most of the important elements and constructs of the operating system, including the CPU and the processes, as objects [62]. The kernel can be partitioned into a set of portable machine independent classes and a comparatively smaller set of non-portable machine dependent classes. This is made possible as a result of widespread use of inheritance. The Choices operating system is internally structured as a hierarchy of objects of various classes. The operating system can be adapted and customized by modifying one or more of these objects. A special browser is available to view the objects being created and used by the operating system in the kernel mode and in the user mode. Choices run on various types of computers including personal computers, supercomputers as well as handheld and mobile devices. Moreover, Choices has been ported to run on several platforms including SPARC, Intel x86 and ARM processors.

4.4. BeOS

BeOS is a hybrid microkernel based operating system [63]. It was development by the Be Inc. BeOS was initially designed and implemented to run on the BeBox platform. However, it was later ported to a plethora of platforms. BeOS was designed and implemented to take maximum advantage of all the novel features of the contemporary computer hardware. Consequently, BeOS benefited from several features of the multiprocessor systems including modular input/output, multithreading and preemptive CPU scheduling. Special emphasis was given to support digital media. The user interface of the BeOS operating system consists of a command line interface that uses the bash shell and a graphical user interface. The GUI of the BeOS is one of the most well designed and well implemented graphical user interface ever developed. The application programmers' interface of the BeOS was written in C++ [64]. Although it is a non-Unix-like operating system, BeOS is POSIX compatible [65].

ISSN 1512-1232

4.5. Syllable

Syllable is an object oriented operating system [66]. It is being developed by a team including Kristian van der Vliet, Kaj de Vos, Rick Caudill, Arno Klenke and Henrik Isaksson. The objective of Syllable is to create an easy to use desktop operating system for home and office users. It is an open source project. Syllable is Unix-like in nature and uses a monolithic kernel. It is mostly POSIX compliant. Syllable is implemented predominantly in C++ and is often compared to the BeOS operating system. Syllable is developed to run on the Intel x86 platform.

4.6. Taj

Taj is an object oriented operating system [67]. It was developed by Viral Patel. Taj is a multitasking and multiuser operating system. It was implemented primarily in C++ with some parts in assembly language. The Taj operating system is implemented as a collection of classes and consequently the source code is highly modular. All the data are highly secured by making them private and removing any chances of unauthorized access. The Taj operating system also facilitates the use of object oriented features like inheritance, and static and dynamic polymorphism. The kernel of the Taj operating system is monolithic in nature. Almost all important components of the operating system including the device drivers are contained inside the kernel to enhance speed.

4.7. Spring

Spring is a microkernel based object oriented operating system [68]. It was developed as an experimental endeavor by the Sun Microsystems. One of the key objectives of the Spring operating system was to provide an enhanced programming support using advanced features like multiple inheritance. The Spring microkernel used several new concepts in implementing interprocess communications, virtual memory and file system. Although the development of the Spring operating system was later abandoned, several ideas and some code from the project were reused in the Java programming language libraries and the Solaris operating system.

4.8. JavaOS

JavaOS is an operating system developed predominantly in Java [69]. It is being developed by the Sun Microsystems. JavaOS uses a Java Virtual Machine. The Java Virtual Machine runs on top of a native microkernel. The device drivers and the important system components of JavaOS are implemented in Java and executed by the virtual machine [70]. The graphics and windowing system of JavaOS, supporting the abstract window toolkit based application programmers' interface, is also implemented in Java. JavaOS has been ported to several platforms including Intel x86, ARM, PowerPC, RISC, SPARC and StrongARM. JavaOS has been developed to be used in various types of embedded systems. It has been employed in several classes of devices including set top boxes, networking devices and JavaStation.

4.9. JNode

JNode is an abbreviation for Java New Operating System Design Effort [71]. It is an open source project to create an operating system using the well established Java platform. Ewout Prangsma is one of the key contributors in this research. This project is being implemented in Java with a minor use of assembly languages. JNode currently supports several file systems, the TCP/IP protocol suite, a graphical user interface and USB peripherals.

4.10 JNUOS

JNUOS is an abbreviation for Jawaharlal Nehru University Operating System [72,73]. The operating system has been developed to serve as a pedagogical tool for the courses on operating systems. It has a microkernel based architecture. Among all microkernel based architectures, the multiple server microkernel based architecture has been used because of its superior modularity

ISSN 1512-1232

[74]. The operating system has a modular and stratified design with five distinct layers comprising of system components and application programs [75]. The operating system has been implemented predominantly in C++. To realize the different constructs of the operating system, twenty-three different classes have been first defined. All important entities in the operating system have been then modeled as objects of these classes [76]. The operating system supports a character user interface and a simple graphical user interface [77]. JNUOS introduces the concept of verbose mode of operation of an operating system [78]. The verbose mode facilitates the users to learn more about the internal working of the operating system. The verbose mode has been implemented using three components, viz., the information server, the verbose server and the explanation module. The verbose server maintains a log of all the important events occurring in the computer system and also collaborates with the information server to create a portrayal of the operation of the entire operating system. This portrayal, or a part of it, is available to any user program on demand. The verbose mode is then completed by the explanation module that probes the verbose server and explains its replies to the users in a stepwise manner. The verbose mode has been tested for providing information of various types and varied depths. The verbose mode typically reports all process related activities including creation, termination and scheduling of all types of processes. Moreover, the output of the explanation module can be customized.

5. Conclusions

In this paper, the pedagogical and research purpose operating systems [79] developed till now have been classified in three categories on an ideological basis. A total of thirty-four such operating systems developed by different organizations have been reviewed briefly. Two significant observations were made. Firstly, the microkernel based architecture is being used much more frequently to develop research purpose operating systems than to develop their commercial counterparts. Secondly, the class of object oriented operating systems, although now in its research stage, is emerging as a promising category of operating systems. So, it is perhaps the right time to amalgamate the concepts of the microkernel based architecture with the object oriented methodologies.

Acknowledgements

Late Prof. R.G. Gupta coauthored the initial draft of this paper. The author also acknowledges Prof. P.C. Saxena, Prof. C.P. Katti and Dr. Anurag Dixit for their guidance and support during the course of this study.

ISSN 1512-1232

References

[1] Silberschatz, A., Galvin, P.B. and Gagne, G., *Operating System Principles*. 7th ed., John Wiley & Sons, 2006.
[2] Tanenbaum, A.S. and Woodhull, A.S., *Operating Systems Design and Implementation*. 3rd ed., Prentice-Hall, 2006.
[3] Hansen, P.B., *Operating System Principles*. Prentice-Hall, 1973.
[4] Chakraborty, P., A model of the computer operating systems. *Journal of Management and Information Technology*, 2009, **1**(1): 83-95.
[5] Madnick, S.E. and Donovan, J.J., *Operating Systems*. McGraw-Hill, 1974.
[6] Rosen, S., Electronic computers: A historical survey. *ACM Computing Surveys*, 1969, **1**(1): 7-36.
[7] Rosin, R.F., Supervisory and monitor systems. *ACM Computing Surveys*, 1969, **1**(1): 37-54.
[8] http://en.wikipedia.org/wiki/Operating_System *Operating System*.
[9] Herder, J.N., *Towards a True Microkernel Operating System*. M.Sc. Dissertation, Vrije Universiteit, Amsterdam, 2005.
[10] Ritchie, D.M. and Thompson, K., The UNIX time sharing system. *Communications of the ACM*, 1974, **17**(7): 365-375.
[11] Bach, M.J., *The Design of the UNIX Operating System*. Prentice-Hall, 1986.
[12] Lions, J., *Lions' Commentary on UNIX 6th Edition with Source Code*. 6th ed., Peer-to-Peer Communications, 1996.
[13] http://www.bell-labs.com/history/unix *The Creation of the UNIX Operating System*.
[14] http://www.minix3.org *MINIX 3*.
[15] Tanenbaum, A.S. and Woodhull, A.S., *Operating Systems Design and Implementation*. 2nd ed., Prentice-Hall, 1997.
[16] http://www.linux.org/docs *Documentation*.
[17] McDougall, R. and Mauro, J., *Solaris Internals: Core Kernel Components*. Prentice-Hall, 2001.
[18] http://www.sun.com/software/solaris *Solaris*.
[19] Comer, D.E., *Operating System Design: The Xinu Approach*. Prentice-Hall, 1984.
[20] Comer, D.E. and Munson, S., *Operating System Design, Vol. 2: Internetworking with XINU*. Prentice-Hall, 1987.
[21] http://www.cs.purdue.edu/research/xinu.html *XINU*.
[22] Pike, R., Presotto, D., Dorward, S., Flandrena, B., Thompson, K., Trickey, H. and Winterbottom, P., Plan 9 from Bell Labs. *Computing Systems*, 1995, **8**(3): 221-254.
[23] http://plan9.bell-labs.com/plan9 *Plan 9 from Bell Labs*.
[24] http://www.vitanuova.com/inferno *A Compact Operating System for Building Cross Platform Distributed Systems*.
[25] Yurkoski, C.F., Rau, L.R. and Ellis, B.K., Using Inferno to execute Java on small devices. *Lecture Notes in Computer Science*, 1998, **1474**: 108-118.
[26] http://www.lsub.org/ls/planb.html *Plan B 4th Edition*.
[27] http://www.sgi.com/products/software/irix/datasheet.pdf *IRIX 6.5*.
[28] http://www.lynuxworks.com/rtos *LynxOS RTOS*.
[29] Dijkstra, E.W., The structure of the "THE" – multiprogramming system. *Communications of the ACM*, 1968, **11**(5): 341-346.
[30] Corbato, F.J. and Vyssotsky, V.A., Introduction and overview of the Multics system. *Proceedings of AFIPS Fall Joint Computer Conference*, 1965, pp. 185-196.
[31] Daley, R.C. and Neumann, P.G., A general purpose file system for secondary storage. *Proceedings of AFIPS Fall Joint Computer Conference*, 1965, pp. 213-229.
[32] David, E.E. and Fano, R.M., Some thoughts about the social implications of accessible computing. *Proceedings of AFIPS Fall Joint Computer Conference*, 1965, pp. 243-247.

ISSN 1512-1232

[33] Glaser, E.L., Couleur, J.F. and Oliver, G.A., System design of a computer for time sharing applications. *Proceedings of AFIPS Fall Joint Computer Conference*, 1965, pp. 197-202.
[34] Ossanna, J.F., Mikus, L. and Dunten, S.D., Communications and input-output switching in a multiplexed computing system. *Proceedings of AFIPS Fall Joint Computer Conference*, 1965, pp. 231-241.
[35] Vyssotsky, V.A., Corbato, F.J. and Graham, R.M., Structure of the Multics supervisor. *Proceedings of AFIPS Fall Joint Computer Conference*, 1965, pp. 203-212.
[36] Organick, E.I., *The MULTICS System: An Examination of Its Structure*. MIT Press, 1972.
[37] http://www.multicians.org *Multics*.
[38] Cheriton, D.R., The V distributed system. *Communications of the ACM*, 1988, **31**(3): 314-333.
[39] http://www.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html *The Mach Project Home Page*.
[40] Rashid, R.F., From RIG to Accent to Mach: The evolution of a network operating system. *Proceedings of ACM/IEEE Computer Society Fall Joint Conference*, 1986, pp. 1128-1137.
[41] Liedtke, J., Toward real microkernels. *Communications of the ACM*, 1996, **39**(9): 70-77.
[42] Hartig, H., Hohmuth, M., Liedtke, J., Schonberg, S. and Wolter, J., The performance of microkernel based systems. *Proceedings of Sixteenth ACM Symposium on Operating System Principles*, 1997, pp. 66-77.
[43] Shapiro, J.S., Smith, J.M. and Farber, D.J., EROS: A fast capability system. *Proceedings of Seventeenth ACM Symposium on Operating Systems Principles*, 1999, pp. 170-185.
[44] Shapiro, J.S. and Hardy, N., EROS: A principle driven operating system from the ground up. *IEEE Software*, 2002, **19**(1): 26-33.
[45] http://www.eros-os.org *EROS: The Extremely Reliable Operating System.*
[46] http://www.capros.org *CapROS: The Capability Based Reliable Operating System*.
[47] http://www.coyotos.org *The Coyotos Secure Operating System*.
[48] http://www.cs.vu.nl/pub/amoeba *The Amoeba Distributed Operating System*.
[49] van Renesse, R., van Staveren, H. and Tanenbaum, A.S., Performance of the Amoeba distributed operating system. *Software – Practice and Experience*, 1989, **19**(3): 223-234.
[50] Tanenbaum, A.S., van Renesse, R., van Staveren, H., Sharp, G.J., Mullender, S.J., Jansen, A.J. and van Rossum, G., Experiences with the Amoeba distributed operating system. *Communications of the ACM*, 1990, **33**(12): 46-63.
[51] Tanenbaum, A.S., *Distributed Operating Systems*. Prentice-Hall, 1995.
[52] http://programatica.cs.pdx.edu/House *Haskell User's Operating System and Environment*.
[53] http://www.rink.nu/trac/ilios *Trac Integrated SCM and Project Management*.
[54] http://www.ghs.com/products/rtos/integrity.html *Integrity Real Time Operating System*.
[55] http://www.cl.cam.ac.uk/research/srg/netos/old-projects/nemesis *Nemesis*.
[56] http://research.microsoft.com/os/singularity *Singularity*.
[57] Hunt, G. and Larus, J., Singularity: Rethinking the software stack. *ACM SIGOPS Operating Systems Review*, 2007, **41**(2): 37-49.
[58] Jones, A.K., The object model: A conceptual tool for structuring software. *Lecture Notes in Computer Science*, 1978, **60**: 7-16.
[59] http://www.nextarchive.net *NeXT Archive*.
[60] http://www.rocklyte.com/athene *The Athene Operating System*.
[61] http://choices.cs.uiuc.edu *Choices*.
[62] Campbell, R.H., Johnston, G., Kenny, K., Murakami, G. and Russo, V., Choices (class hierarchical open interface for custom embedded systems). *ACM SIGOPS Operating Systems Review*, 1987, **21**(3): 9-17.
[63] Hacker, S., Bortman, H. and Herborth, C., *The BeOS Bible*. Peachpit Press, 1999.
[64] Sydow, D.P., *Programming the Be Operating System.* O'Reilly, 1999.
[65] Brown, M.C., *BeOS: Porting UNIX Applications*. Morgan Kaufmann, 1998.
[66] http://www.syllable.org *Syllable*.

ISSN 1512-1232

[67] http://www.viralpatel.net/taj/home.php *TAJ – An Object Oriented Operating System*.
[68] http://research.sun.com/features/tenyears/volcd/papers/Mitchell.pdf *An Overview of the Spring System*.
[69] https://javaos.dev.java.net *JavaOS Project Home*.
[70] Mirho, C.A. and Saulpaugh, T., *Inside the JavaOS Operating System*. Addison-Wesley, 1999.
[71] http://www.jnode.org/node/1 *JNode Handbook*.
[72] Chakraborty, P., *Design and Implementation Considerations for a Pedagogical Object Oriented Operating System*. M.Tech. Dissertation, Jawaharlal Nehru University, 2008.
[73] Chakraborty, P., Thesis overview: Design and implementation considerations for a pedagogical object oriented operating system. *Journal of Computer Science and Technology*, 2010, **10**(1): in press.
[74] Chakraborty, P. and Gupta, R.G., A structural classification and related design issues of operating systems. *Proceedings of Second National Conference on Methods and Models in Computing*, 2007, pp. 265-273.
[75] Chakraborty, P. and Gupta, R.G., The design of a pedagogical operating system. *Proceedings of Second National Conference on Computing for Nation Development*, 2008, pp. 517-527.
[76] Chakraborty, P. and Saxena, P.C., The object model of the JNUOS operating system. *Proceedings of Third National Conference on Computing for Nation Development*, 2009, pp. 281-284.
[77] Chakraborty, P. and Saxena, P.C., A comparison of the shell commands of three contemporary operating systems. *Proceedings of National Conference on Modern Trends in Information Technology*, 2009, pp. 89-95.
[78] Chakraborty, P., Verbose mode of operation of a pedagogical virtual machine operating system. *Proceedings of Third National Conference on Methods and Models in Computing*, 2008, pp. 43-53.
[79] Chakraborty, P. and Saxena, P.C., Novel approaches to teach and learn courses on computer operating systems. *Computer Science and Telecommunications*, 2009, **5**(22): 174-176.

Article received: 2010-03-02