



Resources for Teaching Operating Systems: A Survey of Instructors and a Literature Review

MARIA R. EBLING, United States Military Academy, West Point, NY, USA

Objectives: Faculty new to teaching operating systems or those looking to refresh their course need to understand the state of the art in operating system education. Toward this goal, we conducted a survey of operating system instructors to understand how they approach teaching the course and what textbook and software platforms they use in their classrooms. We also conducted a literature review examining two decades of papers focused on teaching operating systems to undergraduates.

Survey: We surveyed people who teach operating systems at the undergraduate level to determine which textbook they use, on which software they base projects, and how they approach teaching the course.

Literature Review: We searched the ACM Digital Library for publications focused on undergraduate education about operating systems. We selected a total of 51 papers. For each selected paper, we identified which approach (i.e., concrete vs. abstract) and perspective (i.e., internal or external) on teaching operating systems the authors used, which Curriculum 2023 topics they covered, and on what type of system (e.g., educational, production, research) their projects were based. We also looked at the evaluation of the methods and the impact these papers have had on the field.

Findings: Instructors teaching operating systems tend to balance both their approach and their perspective, with a slight leaning toward an internal perspective, whereas authors supporting operating system education overwhelmingly support an internal, concrete approach to the course. In addition, authors also tend to focus on a few key topics, including System Calls and Processes, Concurrency, Scheduling, Virtual Memory, File Systems API and Implementation, and Performance Evaluation.

Conclusions: This work will help faculty teaching operating systems to consider which approach and perspective they wish to take in their course and to find the resources most relevant to their preferred approach and perspective. It can also help focus future research in the area of operating system education, to align it better with current practices.

CCS Concepts: • **Social and professional topics** → **Computing education; Computer science education; Software and its engineering** → *Operating systems*;

Additional Key Words and Phrases: undergraduate, literature review, narrative review

ACM Reference format:

Maria R. Ebling. 2024. Resources for Teaching Operating Systems: A Survey of Instructors and a Literature Review. *ACM Trans. Comput. Educ.* 24, 4, Article 46 (October 2024), 28 pages.

<https://doi.org/10.1145/3688853>

Author's Contact Information: Maria R. Ebling (corresponding author), United States Military Academy, West Point, NY, USA; e-mail: maria.ebling@westpoint.edu.

The views expressed in this article are those of the author and do not reflect the official policy or position of the Department of the Army, Department of Defense, or the U.S. Government.

This paper is authored by an employee(s) of the United States Government and is in the public domain. Non-exclusive copying or redistribution is allowed, provided that the article citation is given and the authors and agency are clearly identified as its source.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 1946-6226/2024/10-ART46

<https://doi.org/10.1145/3688853>

1 Introduction

The OS is the system software that manages the computer's hardware and software resources to provide common services for users and their application-level programs. The OS allows the computer to be shared by many processes and potentially many users. It protects each process from other processes and each user from other users. The code that implements the OS is almost always written in a combination of C and assembly language. OSs represent a challenging body of code to understand, implement, enhance, and debug; they present, therefore, a valuable learning opportunity for **computer science (CS)** students.

This article reports on an investigation looking at questions of how instructors approach teaching OSs at the undergraduate level and what perspective they take. It also seeks to understand ways in which the research literature on OS education supports these approaches and perspectives. Our motivation for asking these questions is to support faculty teaching OSs, particularly new faculty asked to teach the course or faculty tasked with refreshing the course. This article also provides valuable information for those conducting research in the area of OS education.

Our investigation focuses on the following **research questions (RQs)**:

- (RQ1) How do instructors approach teaching OS?
- (RQ2) What textbook(s) do instructors use in OS?
- (RQ3) On what systems do instructors base OS projects/assignments?
- (RQ4) What OS instructional approaches does the CS research literature describe?
- (RQ5) What are the key topics covered in this research literature?
- (RQ6) What types of software platforms are covered in this research literature?
- (RQ7) How do authors evaluate their work?
- (RQ8) What impact have these papers had?
- (RQ9) Who is innovating in the domain of OS instruction?

To answer these questions, we surveyed instructors of OSs and also conducted a literature review of papers that discuss research on OS education. Throughout the remainder of this article, the term *instructor* refers to an educator who responded to the survey. Likewise, the term *author* refers to an author of a paper included in the literature review.

Overall, instructors tend to take a relatively balanced approach to teaching OSs in both approach and perspective. They tend to have students focus on the internals of an OS and they use a balance between concrete and abstract approaches, as defined in Section 2. When building upon an existing OS code base for projects, instructors predominantly use a production OS, but many use an educational OS and some use a research OS. In contrast, authors tend to focus on an internal perspective and a concrete approach. Furthermore, authors tend to focus on a few key topics: System Calls and Processes, Concurrency, Scheduling, Virtual Memory, File Systems API and Implementation, and Performance Evaluation. Authors frequently report on system supports that allow students to extend educational or production OSs, but some have the students build the OS from scratch.

Authors publishing papers about teaching OSs tend to provide limited evaluation of their teaching methods, most commonly subjective perceptions of their students' or of their own experiences. The papers tend to be downloaded relatively frequently but tend not to be well cited. In addition, many links to artifacts are no longer valid, though a few (e.g., 63, 86) have stood the test of time.

The rest of this article is organized as follows: Section 2 looks at methods for conducting a literature review, examines some existing literature reviews, and summarizes the ways prior work has described the different approaches and perspectives instructors take when teaching OSs. Section 3 discusses both the survey we used to collect information from current instructors and the

method we used to conduct this narrative literature review. Section 4 addresses each RQ using the data we collected from both the survey and the literature review. In Section 5, we make high-level observations about the study and present recommendations to both faculty teaching OSs and researchers studying OS education. Section 6 discusses the limitations of the current study and opportunities for future work. Finally, Section 7 summarizes the key findings.

2 Related Work

We examine different types of related work. The first type consists of different methods for conducting a literature review. The second looks at examples of each type of literature review in a number of different topic areas. The final type looks at different ways to approach teaching an OS course.

Many different methods exist for conducting a literature review. Narrative reviews provide an overview of the literature but do not necessarily strive for exhaustive coverage. They typically look at how different researchers have approached a topic over time [90]. A systematic literature review identifies, evaluates, and interprets *all* available research associated with a particular RQ [47]. In a systematic mapping study, the researcher seeks to understand the frequency of publication over time or the places where such research has appeared [62]. These latter approaches aim to be exhaustive. All three of these approaches set search standards, specify inclusion and exclusion criteria, and take a systematic approach to the literature review. For this work, we draw predominantly upon the first approach, performing a narrative review of the educational literature focused on teaching OSs.

Many educators have reviewed the state of computing literature to understand the state of the art of teaching various courses, and they have used different methods to perform their literature reviews. Sharmin examined the computing education literature looking at the use of creativity in introductory CS courses [79] using a narrative review. Prvan and Ožegović [67] did a systematic literature review inspired by Medeiros [56] and content-analysis guided by Mayring [55] to study approaches to teaching computer networking to undergraduates. Švábenský et al. conducted a systematic literature review of the **Special Interest Group on Computer Science Education's (SIGCSE) Technical Symposium (TS) and Innovation and Technology in Computer Science Education (ITiCSE)** conferences to understand the state of the art in cybersecurity education [85], leveraging the methods from both systematic literature reviews [47] and systematic mapping studies [62]. Our work draws inspiration from these papers, especially the coding schemes described by Švábenský et al. [85].

The OS community has looked at different ways to teach OSs. OSs can be taught from either an internal or external perspective [14, 27]. An *internal* perspective focuses on the internals of the OS; students take on the persona of someone developing an OS. An *external* perspective focuses on the functionality provided by an OS; students take on the role of someone using, or developing programs that use, that functionality. We use these terms throughout this investigation to characterize the perspective used by an instructor or supported by a particular paper.

Similarly, Hovemeyer characterized the approaches to teaching OSs as either concrete or abstract [42]. A *concrete* approach is one that focuses on the design and implementation of a realistic OS kernel. An *abstract* approach focuses on algorithms and techniques in isolation or in a simulated environment. We use these terms throughout this investigation to characterize the approach used by an instructor or supported by a particular paper.

In the area of OS education, the work most related to ours, and the only survey of OS education that we found, is that conducted by Anderson and Nguyen in 2005 [2]. They reviewed educational OSs for use in teaching from a concrete, internal perspective. They also reported on a survey of which educational OSs were in use at campuses in the United States (US) by visiting the Web sites of the top 100, US CS schools, determining who taught OSs, and sending the survey to that individual. They examined a number of educational OSs (e.g., [20, 41, 42]), including the language the educational OS uses, the amount of functionality the educational OS provides to students, and

the development environment. Like their work, our work involves both a survey and a literature review. Their survey focused on the top 100 US-based CS schools, whereas our survey polled institutions from across the CS education and OSs research communities via worldwide distribution lists. In addition, our work considers other approaches to teaching OSs beyond a concrete, internal perspective whereas their work examines specific instructional OSs in more detail.

3 Methods

The RQs require different methods to address. We use both a survey of OS instructors and a literature review of OS education papers. In this section, we summarize the survey we used to address the first three RQs (RQ1–3). We then summarize how we conducted the literature review to address the remaining questions (RQ4–9). Finally, we identify which data is used in answering each of the RQs.

3.1 Survey

To gather information about how OSs is taught at post-secondary institutions, we prepared a survey that asked instructors questions about how they teach OSs to undergraduate students. (This work has all necessary institutional approvals, including that of our Institutional Review Board.) The topics included the following:

- (1) the textbook(s) used
- (2) the platform(s) used
- (3) the perspective (internal or external)
- (4) the approach (concrete or abstract)

The first two topics were addressed in the survey by questions that included selection lists. Instructors could select one or more of the textbooks (or platforms) in the list, but also had the option to fill in a textbook (or platform) that was not included in our list and the option to indicate that they do not use a textbook (or platform). The textbooks listed were those by Anderson and Dahlin [3], Arpaci-Dusseau and Arpaci-Dusseau [6], Bach [9], Silberschatz et al. [82], Stallings [84], Tanenbaum and Bos [87], and Tanenbaum and Woodhull [88]. The platforms listed were BabyOS [51], Embedded Xinu [15], FOSL [21], GeekOS [42], iPodLinux [50], Kaya [35], Linux, MiniOS [61], Minux [86], Nachos [20], OS/161 [41], OSTEP Projects [5], PennOS [8], Pintos [63], TOS [57], and xv6 [23]. Many instructors specified textbooks and/or platforms not included in our list.

We also collected data about the instructors and their institutions, including:

- (1) whether they had taught OSs within the past 2 academic years
- (2) the type of institution at which they taught, including whether it is a public or private institution and whether it is a college or university
- (3) the level of undergraduates taking the course (e.g., juniors, seniors)

The exact questions used in the survey can be seen in Appendix A. The survey was created by the author and test driven by colleagues prior to distribution, in addition to the required institutional reviews.

Because we wanted data from recent offerings of OSs, we limited our data to information from the past 2 years. If an instructor indicated that they had not taught OSs within the past 2 years and had given permission to be contacted for clarifications, we reached out to confirm that the data they provided were accurate for an offering within the past 2 years or to request that they forward the survey to the instructor who taught within that time frame. One instructor explained that, although they had not taught the course in the past 2 years, they would be teaching the course in the Fall of 2023 and the data were accurate for this upcoming offering; we included that response. If we were unable to get that confirmation or if the instructor did not give permission to be contacted, we removed their data from the study.

Table 1. Summary of Search and Selection of Articles

Search	Candidates	Reason Eliminated				Selected
		Short	Type	Abstract	Reading	
SIGCSE with CCS	231	17	10	174	5	25
SIGOPS Review	9	1	0	2	1	5
SIGCSE without CCS	91	9	1	67	2	12
SIGOPS Educational	18	0	0	15	0	3
Previously Identified	8	0	0	0	1	7
Historical Context	3	0	0	0	0	3
Remove Duplicates						-4
					Total	51

A few instructors reported that they teach at institutes of technology. We categorized these as either colleges or universities, depending on whether the institution offers a post-graduate degree or not.

A few instructors reported a mixed classroom. For example, one instructor reported having a mix of juniors and seniors and another reported having a mix of sophomores and juniors. In each case, we classified the course with the lower of the options (i.e., juniors and sophomores, respectively). Another instructor reported having second- and third-year students, which we classified as sophomores and juniors. In addition, one instructor reported teaching the course to graduate students; this response was removed from the dataset because our focus is on undergraduate education.

In one case, we received more than one response from the same institution with conflicting information. Because these faculty permitted us to reach out to them for clarification, we learned that multiple instructors teach OSs at that institution and academic freedom permits them to take different approaches. We included both responses in the dataset used in this article.

3.2 Literature Review

We identified papers for this review by searching the **Association of Computing Machinery (ACM)** Digital Library’s Guide to Computing Literature collection, which contains well over three million records. Our initial search and selection took place between 10 and 12 July 2022. We updated our search and selection on 6 June 2023. We used the following parameters:

- Terms: (“operating system”)
- Dates: January 2000–July 2022 and August 2022–June 2023
- Published: *ACM SIGCSE Bulletin* OR *ACM Transactions on Computer Education*
- ACM **Computing Classification System** (CCS 2012): Education OR Information science education OR Computing education programs OR Information technology education OR Computer engineering education OR Computer science education OR Adult education OR Information systems education

The *ACM SIGCSE Bulletin* filter captures papers that appeared in either the SIGCSE TS or the ITiCSE conferences (and represented in Table 1 as “SIGCSE”) both A-ranked conferences by Resurichify.¹ This search identified 231 papers.

Next, we expanded this search to also include anything published in *ACM Special Interest Group in Operating Systems (SIGOPS) Operating Systems Review* during the same time frame with the

¹<https://www.resurichify.com/>

same set of CCS 2012 topics. We reasoned that some OS researchers focus more on the OS research community than on the CS education research community. This search identified a total of nine additional papers.

We then eliminated any papers that were either one or two pages long. These short papers typically represent panels, posters, or demonstrations and do not contain sufficient detail to answer our RQs. We removed 17 from the SIGCSE search and 1 from the SIGOPS search. We then eliminated 10 ITiCSE working group reports and guest editorials, none of which were focused on OSs. We then read the titles and abstracts to determine each paper's relevance to teaching OSs to undergraduate CS majors. We eliminated 174 papers from the SIGCSE search and 2 from the SIGOPS search because their focus was not relevant to this review, as shown under the heading "Abstract" in Table 1.

Surprisingly, we discovered that many papers with which we were familiar were not contained in the resulting list and realized that the CCS 2012 parameter was the likely culprit. Indeed, repeating our initial search without the CCS 2012 parameter identified another 91 candidate papers, of which 14 passed our selection criteria. We could not take the same approach on the *ACM SIGOPS Operating Systems Review* search because every paper would have likely been identified. Instead, we added another search in *ACM SIGOPS Operating Systems Review* to look for the term "educational" over the same periods of time. This additional search ("SIGOPS Educational" in Table 1) identified 18 new papers, 3 of which passed our selection criteria, all of which had been identified in a prior search. Even after performing these additional searches, we still had eight papers, with which we were familiar from earlier, less systematic searches of the literature, that did not appear in any of the searches but were published in SIGCSE after 2000 and have the term "operating systems" in both the title and body of the paper. We manually added these papers to our list. We also added three well-known papers for historical reasons, even though they do not meet the search criteria.

At this point, we read each paper confirming its relevance to our search. We eliminated nine additional papers, eight because they were not reporting on a course focused on OSs and one because it was focused on a master's-level course on OSs. These are shown under the "Reading" heading in Table 1.

At the conclusion of our search and filtering, we identified a total of 51 papers for review. Table 1 summarizes the number of candidate papers identified, the number of papers eliminated for each reason, and the number of papers selected in each step during the search.

3.3 Data Extraction

For each RQ, we identified the data required to answer it as follows:

- (RQ1) *How do instructors approach teaching OS?* To answer this question, we rely on the survey of faculty members who teach OSs. These faculty members were drawn from members of the SIGCSE-Members and SIGOPS-review mailing lists. Only the data from those instructors who reported having taught OS to undergraduates in the past 2 years (or the upcoming semester) were included in our analysis.
- (RQ2) *What textbook(s) do instructors use in OS?* To answer this question, we again rely on the responses from faculty who teach OSs.
- (RQ3) *On what systems do instructors base OS projects/assignments?* To answer this question, we again rely on the responses from faculty who teach OSs.
- (RQ4) *What OS instructional approaches does the CS research literature describe?* To answer this question, we turn to our literature review. For each paper, we classified each approach as concrete vs. abstract using a 5-point Likert scale from fully concrete to fully abstract. We classified the perspective the paper takes as internal vs. external on a similar scale from entirely internal to entirely external.

- (RQ5) *What are the key topics covered in this research literature?* To address this question, we again used our literature review. In this case, we identified topics to extract from the selected papers using the Knowledge Units of the Curriculum 2023 Guidelines for Operating Systems [48]. The topics identified were System calls and Processes (part of the Principles of Operating Systems Knowledge and Process Model Units), Concurrency, Scheduling, Memory Management, Virtual Memory (part of the Virtualization Knowledge Unit), Protection and Safety, Device Management, File Systems API and Implementation, Real Time and Embedded Systems, Fault Tolerance, and Performance Evaluation (part of the Foundations of Systems Knowledge Area).
- (RQ6) *What types of software platforms are covered in this research literature?* For each paper, we classified the systems on which the projects were based as educational, production, from scratch, or simulations.
- (RQ7) *How do authors evaluate their work?* To address this question, we classified the type of evaluation, the data collected, and the **analysis methods (AMs)** employed using a coding scheme based on that of Švábenský et al. [85].
- (RQ8) *What impact have these papers had?* For each paper, we noted any artifacts made available to other instructors and also recorded the number of citations and downloads of the paper as reported by the ACM Digital Library.
- (RQ9) *Who is innovating in the domain of OS instruction?* For each paper, we recorded each author and each institution. For example, a paper co-authored by two people will count as one paper for each of those authors (similarly, for institutions).

4 Results

In this section, we present the results of our study. We begin with a quick overview of the survey responses and then provide a quick note about the literature review. For the remainder of this section, we discuss each of the RQs identified in Section 1.

Survey. A total of 66 instructors completed the survey and met the criteria for inclusion in this study. We found that 28 instructors (42.4%) teach at private institutions and 38 instructors (57.6%) teach at public institutions. We also found that 53 (80.3%) teach at universities and 13 (19.7%) teach at 4-year colleges. The majority of students take OSs as juniors (50%) or seniors (28.8%), but some take it as sophomores (19.7%). Two instructors reported a mixed classroom: one with juniors and seniors (mixed equally) and one with sophomores and juniors. In each case, we categorized the data with the lower of the two populations (i.e., juniors and sophomores, respectively) for the purpose of our analysis.

Literature Review. Our literature review identified 51 papers. In the process of analyzing these papers, we found that one paper focused entirely on the order of topics covered in an OSs class and did not address our RQs. As such, it has been omitted from the classifications in the remainder of this article. Readers interested in understanding how one author serialized the many interrelated topics typical in an OS course are referred to Bovet's paper [13]. Thus, for the remainder of this article, we consider 50 papers.

4.1 RQ1: How Do Instructors Approach Teaching OS?

In our survey of OSs instructors, we asked whether instructors approached their course from an internal perspective (with a focus on the implementation of the OS) or from an external perspective (with a focus on using the services an OS provides). We found that instructors lean slightly toward an internal approach, with 40% taking either an entirely internal or mostly internal perspective. The remaining instructors split relatively evenly between an external perspective and a balanced

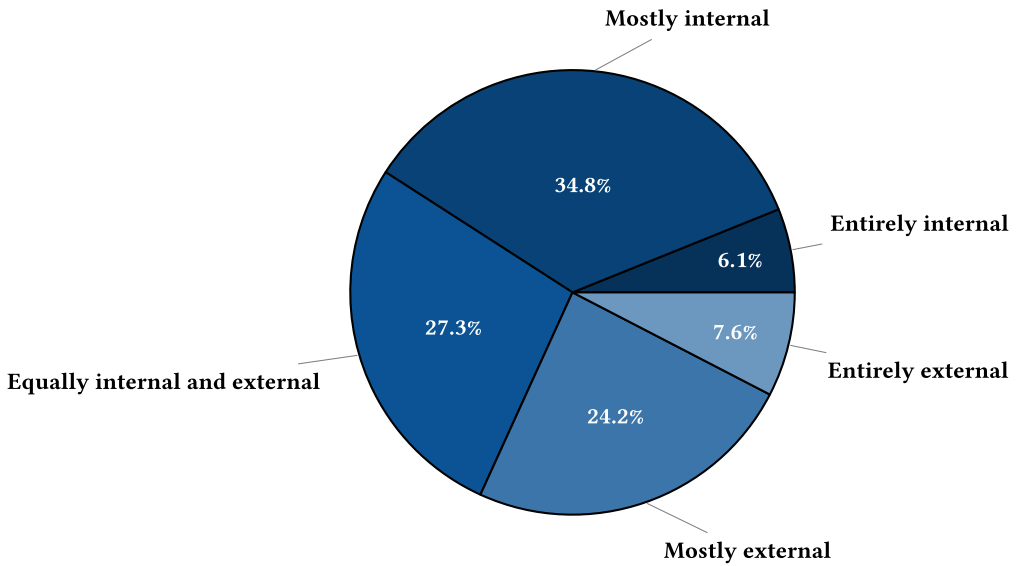


Fig. 1. Teaching from an internal vs. external perspective (n = 66).

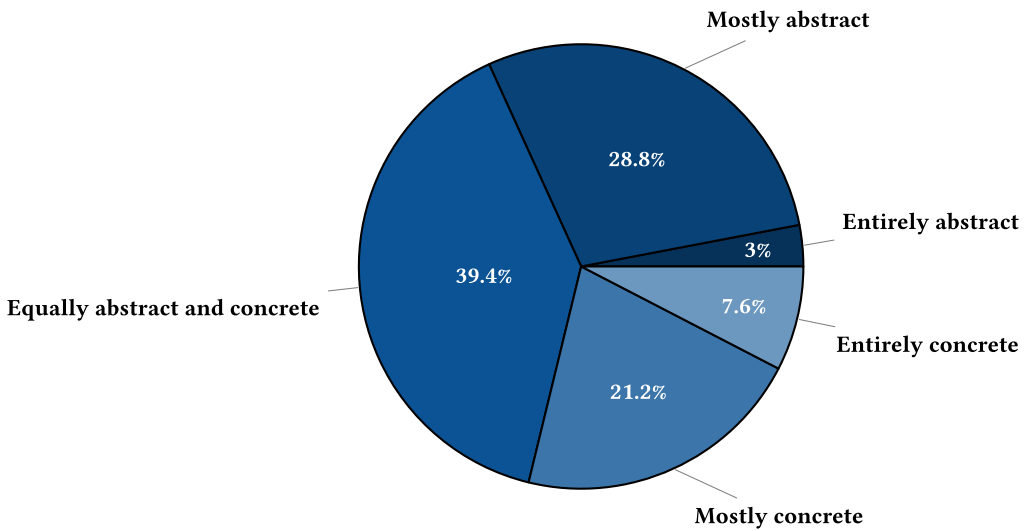


Fig. 2. Teaching from an abstract vs. concrete approach (n = 66).

perspective. Few instructors take either an entirely internal or an entirely external approach. Details are shown in Figure 1.

We also asked whether instructors teach their course using a concrete approach that focuses on the design and implementation of a realistic OS, or using an abstract approach that focuses on algorithms and techniques in isolation or in a simulated environment. We found that almost 40% of instructors take a balanced approach that is equally concrete and abstract. Of the remaining, roughly half lean toward abstract and half lean toward concrete. Once again, few instructors take either an entirely concrete or an entirely abstract approach. The results are shown in Figure 2.

Table 2. Approaches to Teaching OSs in the Survey (n = 66)

	Lean Concrete	Equally Concrete and Abstract	Lean Abstract	Total
Lean Internal	12	7	8	27
Equally Internal and External	4	9	5	18
Lean External	3	10	8	21
Total	19	26	21	66

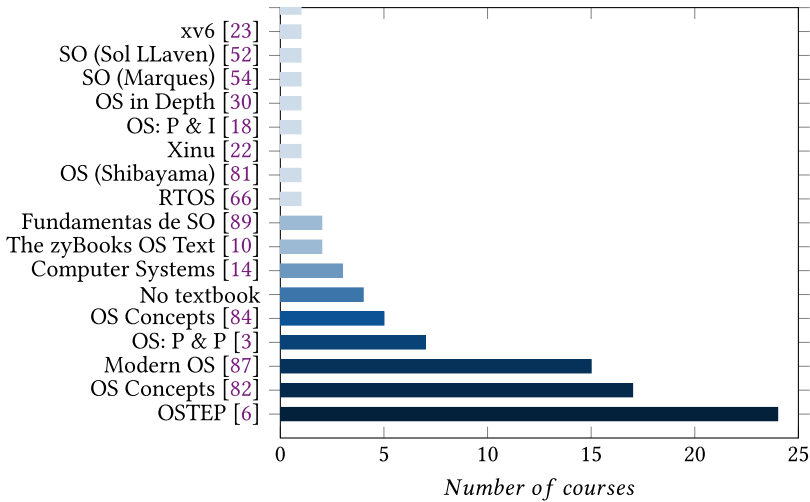


Fig. 3. Summary of textbooks used (n = 88).

Next, we wanted to see how the different perspectives and approaches interact. For this analysis, we collapsed the “mostly” and “entirely” categories into a “leans” category. For example, we considered an instructor who takes an entirely concrete approach and another instructor who takes a mostly concrete approach as both leaning toward a concrete approach. Table 2 shows a summary of how instructors approach teaching OSs and which perspective they take. From this table, it is apparent that instructors favor an internal perspective and a balance between a concrete and an abstract approach.

4.2 RQ2: What Textbook(s) Do Instructors Use in OS?

The survey asked instructors about the textbook(s) they use. Figure 3 shows the textbooks that instructors reported using and the number of instructors who use each. Note that the survey allowed instructors to specify textbooks that we omitted from the list provided in the question via a write-in response. The total number of textbooks reported exceeds the total number of instructors because some instructors use multiple textbooks. When an instructor reported using two textbooks, we marked both as being used. For example, one instructor reported using *Operating Systems: Three Easy Pieces* [6] and supplementing it with *Computer Systems: A Programmers Perspective* [14]. For the purposes of this analysis, each book was marked as being used once from this single response. The most frequently used textbook is *Operating Systems: Three Easy Pieces* by Arpaci-Dusseau

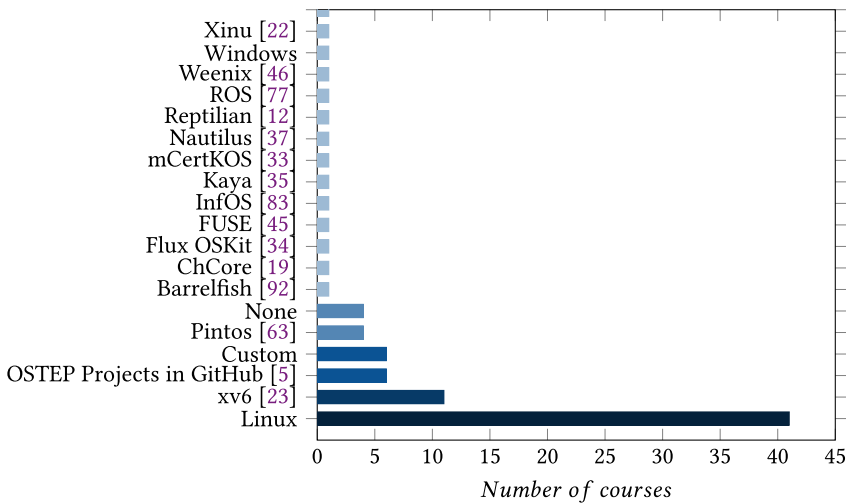


Fig. 4. Summary of platforms used (n = 85).

and Arpaci-Dusseau. *Operating System Concepts* [82] by Silberschatz et al. and *Modern Operating Systems* [87] by Tanenbaum and Bos are the next most common. Details are shown in Figure 3.

4.3 RQ3: On What Systems Do Instructors Base OS Projects/Assignments?

The survey also asked instructors about what software platforms they use to teach OSs. Figure 4 shows the details of what platforms were identified. Note that the survey allowed instructors to specify platforms that we omitted from the list provided in the question via a write-in response. Once again, the total number of platforms reported exceeds the total number of instructors because some instructors use multiple platforms. We then categorized the responses to examine, at a high level, the types of platform in use. For this purpose, Linux, Windows, and FUSE were categorized as production. Similarly, barrelfish [92], mCertKOS [33], Nautilus [36, 37], and ROS [77] were categorized as research. All other platforms were categorized as educational.

Next, we looked at whether there are differences in the type of platform chosen based upon the institution’s category: private vs. public and college vs. university. Production OSs are more commonly used than educational OSs across all categories. Table 3 shows these data.

Next, we examined how the basis for projects has changed over time by comparing with the Anderson and Nguyen survey from 2005 [3]. In Table 4, we show all of the platforms included in the Anderson and Nguyen survey as well as those used by more than one of our instructors. Recall that the “Unix-oriented” and “Java-based without an OS” responses from Anderson and Nguyen involved using the Unix APIs and using the Java language to explore OS concepts, without using an instructional OS. As is clear from this table, instructors have made a notable shift to Linux and xv6.

4.4 RQ4: What OS Instructional Approaches Does the CS Research Literature Describe?

We now look to the CS literature supporting OS education and which approaches to OS education the literature supports. For each of the selected papers, we evaluated whether the paper takes a concrete or abstract approach and whether it takes an internal or external perspective. In some cases, the approach and/or perspective included aspects of both concrete and abstract or both internal and external; in this case, we marked these as a mixed approach or perspective. Table 5 identifies the approach and perspective taken by each paper. One paper, a tool paper [25], advocates

Table 3. Project Basis by Institution (n = 66)

Basis	#	Institution Types			
		Private	Public	College	University
Educational OS	11	5	6	1	10
Production OS	29	12	17	7	22
Both Production and Educational	13	5	8	3	10
Research OS	3	2	1	0	3
Both Production and Research	1	0	1	0	1
Custom OS	3	0	3	0	3
Simulations	1	0	1	0	1
No Platform	5	4	1	2	3
Total	66	28	38	13	53

Table 4. Historical Comparison

Project Basis	2005 Usage	2023 Usage
GeekOS	3%	0%
Java-based without an OS	8%	0%
JOS	3%	0%
Linux	14%	62%
Minix	7%	0%
Nachos	17%	0%
OS/161	5%	0%
Pintos	N/A	6%
Unix-oriented	27%	0%
Xinu	4%	0%
xv6	N/A	17%
Yalnx	3%	0%

a tool that supports providing instructor comments in code. Because this tool could be used from either an internal or external perspective and with either a concrete or abstract approach, we did not include it in Table 5; therefore, n = 49 for this table.

As is evident in Table 5, most of the selected papers discuss teaching OSs to undergraduate students using a concrete approach from an internal perspective. This combination is more than three times as frequent as any other option.

4.5 RQ5: What Are the Key Topics Covered in This Research Literature?

For each assignment discussed in a paper, we identified the Curriculum 2023 [48] topic with which it was most closely associated. Curriculum 2023 further breaks down topics as CS Core and **Knowledge Area (KA) Core**. CS Core topics are those that *every* CS graduate should know. A program may, however, choose to cover some KA Core topics in more depth at the expense of other KA Core topics. The analysis presented here does not delve into the categorization of topics to the level of CS Core and KA Core.

Table 5. Approaches to Teaching Operating Systems in the Literature (n = 49)

	Concrete (n = 33)	Mixed (n = 1)	Abstract (n = 15)
Internal (n = 35)	[1, 4, 7, 11, 15, 20, 21, 23, 28, 29, 31, 32, 35, 39, 41–43, 49–51, 58, 60, 61, 63, 86, 91]	[65]	[40, 53, 59, 70, 73–76]
Mixed (n = 6)	[8, 57, 64, 78]		[16, 71]
External (n = 8)	[17, 26, 80]		[38, 44, 68, 69, 72]

A few points are worth noting:

- The *System Calls and Processes* topic covers aspects of both the Principles of Operating Systems and the Process Model Knowledge Units within the Operating Systems Knowledge Area. These two topics are, however, frequently used together as an introductory topic in OSs courses so they are combined in this article for convenience.
- The *Virtual Memory* topic is formally part of the *Virtualization* Knowledge Unit within the Operating Systems Knowledge Area. It is called out here only as Virtual Memory due to the specificity of the topic within the literature.
- The *Performance Evaluation* topic is formally part of the Fundamentals of Systems Knowledge Area, but is a skill frequently taught as part of a classic OS course so is included here for convenience.

A summary of the topic areas can be seen in Table 6. For each topic, we show the total number of papers that included this topic as well as the citations to the specific papers. One paper included in Table 5 does not describe any projects; therefore, with the omission of the previously mentioned tool paper and this additional paper, n = 48 in Table 6.

We observed six topics that are widely covered, sometimes in similar ways and sometimes in quite different ways. These were System Calls and Processes, Concurrency, Scheduling, File System API and Implementation, Virtual Memory, and Performance Evaluation. Memory Management and Device Management are also fairly common topics in the educational literature. The remaining topics, Protection and Safety, Real Time and Embedded Systems, and Fault Tolerance, are uncommon.

Next, we wanted to understand which systems discussed in the research literature have projects related to topics from Curriculum 2023 [48]. In Table 7, you can see which papers have projects related to which topics. We use the following notation:

- ✓ - Means the topic is covered
- † - Means the topic is discussed, but details are omitted
- o - Means the assignment was optional

The Knowledge Units on advance file systems, fault tolerance, and society, ethics, and the profession are not well covered in the literature and are omitted from the table to improve readability.

4.6 RQ6: What Types of Software Platforms Are Covered in This Research Literature?

A total of 43 papers describe assignments or projects to help students learn about OSs. For these papers, we recorded the platform on which the projects (or assignments) are based using the following coding scheme, where “PB” stands for “Project Basis”:

Table 6. Topics Summary (n = 48)

Topic	#	Papers
System Calls and Processes	26	[4, 8, 11, 15, 20, 21, 23, 29, 31, 35, 39–43, 49, 50, 53, 58, 61, 63, 65, 75, 78, 86, 91]
Concurrency	32	[4, 7, 15–17, 20, 21, 23, 29, 31, 32, 35, 38, 39, 41–43, 49, 51, 57, 58, 61, 65, 68–72, 75, 76, 80, 91]
Scheduling	28	[4, 7, 8, 15, 21, 23, 28, 29, 31, 32, 35, 39, 41–43, 49, 51, 53, 58, 60, 61, 63, 65, 73, 76, 80, 86, 91]
Memory Management	12	[15, 20, 23, 31, 39, 51, 60, 61, 64, 65, 86, 91]
Virtual Memory	17	[4, 11, 20, 23, 29, 35, 41–43, 49, 53, 58, 59, 63, 74, 76, 78]
Protection and Safety	3	[26, 44, 65]
Device Management	14	[15, 21, 23, 26, 28, 29, 35, 39, 57, 61, 65, 73, 86, 91]
File Systems API and Implementation	25	[4, 7, 8, 11, 15, 20, 23, 28, 29, 35, 39, 41–43, 49, 50, 57, 58, 60, 61, 63, 65, 76, 80, 86]
Real Time and Embedded Systems	3	[15, 61, 65]
Fault Tolerance	1	[65]
Performance Evaluation	17	[20, 21, 28, 39, 41, 42, 51, 60, 61, 65, 70, 71, 73, 76, 78, 80, 91]

- (PB1) *Educational OSs* are OSs written to support the educational needs of an undergraduate OSs course (e.g., Pintos, xv6).
- (PB2) *Production OSs* are OSs used in production (e.g., Linux, Windows).
- (PB3) *From Scratch* are typically implemented entirely by the student during the semester.
- (PB4) *Simulations* are used to simulate different aspects of an OS, such as a scheduler or address translation.
- (PB5) *Topic-Specific Tools* provide support for teaching a specific OS concept.
- (PB6) *Service Learning* takes students into the field to provide OS support to a non-profit entity.

Finally, we examined whether the basis for projects has shifted over time. For each paper, we identified its year of publication and the high-level basis for its projects (e.g., educational, production). These results are shown in Table 8. We found that the majority of papers propose using educational (12) or production (8) OSs and that many have students build from scratch (5), with what appears to be a trend toward production systems, and away from educational OSs, in the last decade. Many papers reported using simulations (8) or tools (8) to focus on specific topics. One paper uses service learning. The remaining seven papers were not readily categorized, for example because they presented a tool rather than an assignment or project. Thus, $n = 43$ for this analysis.

4.7 RQ7: How Do Authors Evaluate Their Work?

We looked at the **evaluation goals (EG)** using a coding scheme (based upon [85]) where:

- (EG1) *No evaluation*
- (EG2) *Subjective perception* of students or instructors

Table 7. Topics Covered

			Curriculum 2023 Topics									
			SysCalls & Processes	Concurrency	Scheduling	Memory Management	Virtual Memory	Protection & Safety	Device Management	File Systems API & Impl.	Real Time & Emb. Sys	Performance Evaluation
Production	Android	2012	[4]	✓	✓	✓	✓			✓		
	FOSL	2001	[21]	✓	✓	✓			✓			✓
	User-Mode	2004	[26]					✓	✓			
	Linux	2010	[39]	✓	†	✓	✓		✓	+		✓
	Linux	2011	[49]	✓	✓	✓	✓			✓		
	iPodLinux	2008	[50]	✓						+		
	Linux	2005	[58]	✓	✓	✓	✓			✓		
	Windows	2010	[78]	✓				✓				
Educational	PortOS	2002	[7]		✓	✓				✓		
	Emb. Xinu	2008	[15]	✓	✓	✓	✓		✓	✓	✓	
	Nachos	1993	[20]	✓	✓		✓	✓		✓		✓
	xv6	2021	[23]	✓	✓	✓	✓	✓	✓	✓		
	OS/161	2002	[41]	✓	✓	✓	✓	✓		✓		✓
	GeekOS	2004	[42]	✓	✓	✓	✓	✓		✓		✓
	Minix Rev.	2002	[43]	✓	✓	✓		✓		✓		
	BabyOS	2007	[51]		✓	✓	✓					✓
	TOS	2001	[57]		✓				✓	✓		
	Pintos	2009	[63]	✓	✓	✓		✓		✓		
	Minix	1987	[86]	✓		✓	✓		✓	✓		
	Xipx	2013	[91]	✓	✓	✓	✓		✓			✓
	Scratch	Scratch	2009	[11]	✓				o		✓	
HW Sim OS		2000	[29]	✓	✓	✓		✓	✓	✓		
Bottom Up		2001	[31]	✓	✓	✓	✓					
Kaya		2005	[35]	✓	✓	✓		✓	✓	✓		
MiniOS		2015	[61]	✓	o	✓	o		o	✓	✓	o
Simulation	PennOS	2012	[8]	✓	✓	✓				✓		
	Disk Sim	2006	[28]			✓			✓	✓		✓
	Sosim	2005	[53]	✓		✓		✓				
	Bound Buf	2000	[70]		✓							✓
	Monitors	2001	[71]		✓							✓
	Disk Sch	2004	[73]			✓			✓			✓
	Addr Xlation	2005	[74]					✓				
	Concur. I/O	2006	[75]	✓	✓							
	Many	2008	[76]		✓	✓		✓		✓		✓

Table 8. Project Basis (n = 43)

Basis	#	Papers by Year of Publication	
		pre-2010 (n = 35)	2010 or Later (n = 8)
Educational OS	12	[7, 15, 20, 41–43, 51, 57, 63, 86]	[23, 91]
Production OS	8	[21, 26, 50, 58]	[4, 39, 49, 78]
From Scratch	5	[11, 29, 31, 35]	[61]
Simulation	9	[28, 53, 70, 71, 73–76]	[8]
Topic-Specific Tool	8	[16, 17, 32, 38, 40, 68, 69, 72]	
Service Learning	1	[44]	

(EG3) *Objective learning* of students, based upon graded events

(EG4) *Artifacts* produced by students, such as assignments

We found that half of the papers (24) simply reported the subjective perceptions of the students and/or the instructors and nearly half (19) reported no evaluation results whatsoever. The remaining papers were a mix between looking at objective learning measures (4) and examining student-generated artifacts (3).

We also examined at the **types of data (DT)** collected, again using a coding scheme (based upon [85]) where:

(DT1) *No data* were collected

(DT2) *Questionnaire data* about students' subjective perceptions (from EG2)

(DT3) *Graded-event data* that use student grades to evaluate learning

(DT4) *Student artifacts*, such as assignments or exams

(DT5) *Interviews* that examine participants' subjective perceptions

(DT6) *Anecdotal only*, such as from course feedback forms

More than a third (20) of the papers had no data (DT1) and another 14 had only anecdotal data (DT6). Eight papers used a questionnaire (DT2) while 4 assessed student learning from graded events (DT3) and 3 evaluated student artifacts (DT4). One paper used interviews (DT5).

Finally, we examined the AM used, using the coding scheme (from [85]) where:

(AM1) *No analysis* performed

(AM2) *Descriptive statistics*, interpreting data, and reporting means/standard deviations

(AM3) *Inferential statistics*, formulating hypothesis and determining whether it was supported

(AM4) *Qualitative analysis*, expert reviews of student and instructor artifacts.

The majority of papers (36) used no AM (AM1). Of the remaining, all (14) used descriptive statistics (AM2). None used inferential statistics (AM3) or qualitative analysis (AM4).

4.8 RQ8: What Impact Have These Papers Had?

Next, we examined the impact of these papers. We looked at three metrics. First, we looked at the availability of artifacts for instructor reuse. Then we looked at the number of citations and downloads as reported by the ACM Digital Library.

We began by examining how many papers reported the availability of artifacts that could be used by other instructors and whether those papers remain available today. Admittedly, any artifacts that

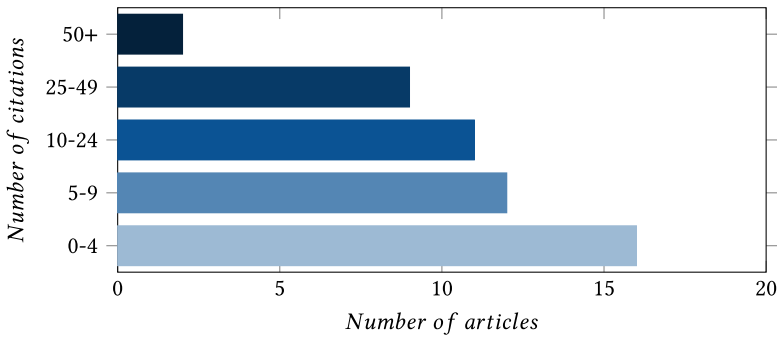


Fig. 5. Citation summary (n = 50).

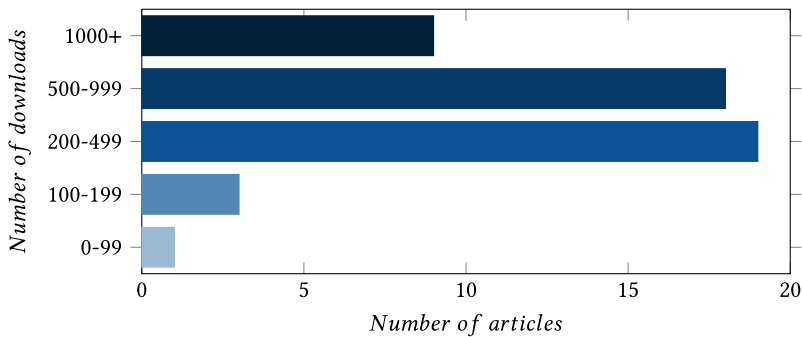


Fig. 6. Download summary (n = 50).

are not available today may still be available through the Internet Archive²; however, if they are not readily available today, they are likely not supported or actively used so we did not investigate their availability in the archive. Of the 31 papers that reported artifacts available for other instructors to use, only 12 of those remain available online today either through the URL provided in the original paper or via a simple internet search, but most of these appear to be inactive and no longer supported. Minux [86], Pintos [63], and xv6 [23] are notable exceptions.

We then recorded the number of citations reported by the ACM Digital Library on 6 July 2023. The number of citations ranges from 1 to 90 with a mean of 15 and a standard deviation of 18. We then categorized each paper by the number of citations into groups of 0–4, 5–9, 10–24, 25–49, and 50 or more citations, as shown in Figure 5. As is evident, many papers are not well cited.

Finally, we recorded the number of downloads reported by the ACM Digital Library on 6 July 2023. The number of downloads ranges from 37 to 2,086 with a mean of 630 and a standard deviation of 423. We categorized each paper by the number of downloads into groups of 0–99, 100–199, 200–499, 500–999, or more than 1,000, as shown in Figure 6. Although there is a high deviation in these numbers, all but four papers had at least 200 downloads. Considering the limited number of people with interest in reading about teaching OSs, we conclude that these papers are reasonably well disseminated.

²archive.org

4.9 RQ9: Who Is Innovating in the Domain of OS Instruction?

We analyzed the authors of the selected papers. There were a total of 88 unique authors across the 50 selected papers. Of these, 78 had authored a single paper, 8 had authored two papers, and 2 had authored more than two papers. Nieh has a series of four papers presenting projects to support teaching OSs from a concrete, internal perspective [4, 25, 49, 58]. Robbins has a series of seven simulation papers that met our selection criteria [70, 76] that support teaching OSs using an abstract approach. Brylow [15, 91], Carr [16, 17], Shene [16, 17], Davoli [26, 35], Donaldson [31, 32], Polze [65, 78], Probert [65, 78], and Reek [68, 69] each authored two papers. We also analyzed the institutions producing these papers, but the analysis was unsurprising with Columbia (Nieh's institution) and University of Texas San Antonio (Robbins' institution) being the only institutions to have more than two papers. In addition, Marquette University, Michigan Technological University, Microsoft Corporation, Oberlin College, and Rochester Institute of Technology each had two papers.

5 Discussion

In this section, we reflect on the study as a whole and the key takeaways. We identify high-level trends and include suggestions for areas our field needs to address. Finally, we make recommendations to both instructors and researchers.

5.1 Production OS Trend

When comparing our instructors' responses to those from Anderson and Nguyen [3], we found an apparent shift toward the use of Linux in undergraduate OSs. In 2005, the use of Linux in OS classrooms stood at 14%, but climbed to 62% by 2023 (see Table 4). This trend is further supported by our literature review. In the 2000s, many publications focused on describing educational OSs, whereas in the 2010s, we saw fewer papers with a focus on educational OSs (see Table 8). At this point, the majority of faculty reported using production OSs as the basis for course projects, though educational OSs still remain in common use.

An advantage of using a production OS is that students immediately see the relevance. The biggest disadvantage, though, is the size of production systems. According to Wikipedia, Linux has more than 30 million lines of code, though only about 14% of this code is the core kernel. An educational OS is a mere fraction of this, with Pintos coming in at approximately 16,000 lines of code and xv6 coming in at approximately 11,000. It is notable that many instructors use both a production OS and an educational OS, likely starting with the smaller, more consumable educational OS and transitioning to the production OS once students become more comfortable with kernel code.

5.2 Artifact Lifetime

As noted earlier, many artifacts made available in one of the papers included in this literature review were unavailable less than 20 years after their publication date. Yet many of the topics taught in OSs have stood the test of time. Our department is undergoing some renovations, and we found bound "course diaries" from the 80s and 90s that included all of the midterm and final exams. Many of the exam questions were ones that could very reasonably be used in a modern OSs course, almost 35 years later. In light of this experience, it is particularly disheartening that so many artifacts described in the literature no more than 20 years ago have already been lost. This observation motivates the need for a repository, such as the EngageCSEdu effort [24]. Of course, even repositories like this will not address the problem of "bit rot."

5.3 Instructor Approaches Inconsistent with Literature

Recall the approaches and perspectives taken by instructors, as shown in Table 2. Most instructors lean toward an internal perspective and an approach balanced between concrete and abstract. Yet, as shown in Table 5, the research literature has predominantly focused on instructors using an internal perspective and a concrete approach.

This mismatch may result from the different backgrounds of instructors. In our experience, “systems people” frequently believe in building systems and will very likely adopt an internal perspective with a concrete approach. We believe much of the literature originates from people with a “systems background” building tools that support the way *they* teach OSs (i.e., from an internal perspective and a concrete approach) and then publishing papers on those tools. In contrast, instructors who do not have a “systems background” may not feel comfortable taking a concrete approach, so they may lean more toward an abstract approach. Even instructors who themselves have a “systems background” may have determined that their students may benefit from a more abstract approach due to where/when the course falls within their curriculum or due to the background of their student population. Regardless of the reason, this observation suggests that the OS community and the OS education research community may want to increase efforts to support instructors wanting to use an abstract approach.

5.4 Recommendations for Instructors

We consider how these results can be applied by instructors new to teaching an undergraduate course in OSs or by instructors who are refreshing an existing course. In planning such a course, a key decision is how to approach the topic: (1) whether to use an internal perspective (focusing on the implementation of the OS itself) or an external one (focusing on using the services an OS provides); (2) whether to use a concrete approach (focusing on the design and implementation of a realistic OS) or an abstract approach (focusing on algorithms and techniques in isolation). A key issue is which perspective and which approach will be most beneficial to the students enrolled in the course.

Another key issue is which knowledge units from Curriculum 2023 [48] to cover and which resources are most applicable to the envisioned course. Instructors need to keep in mind that most resources support a concrete, internal approach and that resources to support an abstract approach are much more limited (at least today). From the survey, we note that instructors tend to avoid the extremes with fewer than 10% choosing to design their class to be either entirely internal/external or entirely abstract/concrete. Once that decision has been made, recall that the Bovet paper [13] focuses on the order of topics in an OS course. Although Bovet’s work substantially predates Curriculum 2023 [48], its reasoning still applies. The information contained in Figure 4 and in Tables 5, 6, and 7 will help identify available resources applicable to these choices. Finally, the top three textbooks used by instructors, shown in Figure 3, are all excellent textbooks, though one benefit of the Arpaci-Dusseau book is that it is a free, online text, reducing the financial burden on students.

5.5 Recommendations for Researchers

We now consider how these results can be applied by CS education researchers focused on supporting an undergraduate course in OSs. As discussed in Section 5.3 above, we found that the vast majority of research reported in the area of OS education supports a concrete, internal approach and yet instructors tend to lean more toward an abstract approach. Future researchers should consider support for these educators to help fill this gap. We also found that the vast majority of research reported in the area of CS education in support of an undergraduate course in OSs has had

little formal evaluation. As is evident in Table 8, most of the research in this literature review was reported prior to 2010. Evidentiary standards have changed substantially over the years. Future research should include more rigorous evaluations.

Unsurprisingly, many of the artifacts reported in this literature review have been lost to the ravages of time. This issue is, admittedly, not limited to the field of CS education. Ensuring artifacts reported in published papers are available is something we need to improve. As new resources are shared with the community, it is critical that steps are taken to ensure their continued availability, even a decade or two after publication.

Finally, we sound a warning. Figures 5 and 6 provide insight about impact in OS education research. These papers are not frequently cited, with very few papers achieving more than 50 citations. They are, however, frequently downloaded with most papers having more than 200 downloads. If you plan to do research in this area as part of a goal to achieve tenure, be aware that metrics focused on citation counts for these publications will likely suffer. In addition, we found that few people publish prolifically in this area. OS education research is not an area of investigation in which researchers tend to specialize.

6 Limitations and Future Work

The first limitation of this study is its scope. The scope is limited to a few publication venues: the ACM SIGSCSE and ITiCSE conferences, the *ACM Transactions on Computing Education* journal, and the *ACM SIGOPS Operating Systems Review*. Despite this limitation, SIGSCSE and ITiCSE are the premiere conferences for reporting on educational research in CS and *ACM SIGOPS Operating Systems Review* supports the special interest group focused on OSs research. Although we would not expect the results to differ substantially if the search were to be expanded, one area for future work would be to extend our literature review to additional publication venues, such as *IEEE Frontiers in Education* and USENIX.

The second limitation is related to the ACM Digital Library (or our use of it). As mentioned in Section 2, our digital library search did not identify eight papers that we would have expected it to identify. The omission of these known papers raises concerns about other papers that might have also been omitted and with which we were not previously familiar. Another area for future work is to investigate further why known papers were not identified, despite numerous search attempts and, once resolved, identify any additional papers that were missed.

The third limitation relates to the survey. The survey of OS instructors presented in this article was done at a very high level and was limited to factual information gathering (e.g., which textbook, which platform). An area for future work would be to do a more detailed survey of OSs instructors to gather additional information. Such a survey could collect more information about the instructors' background (such as demographic data, experience, and country), further information about their assignments and projects, as well as their opinions and experiences. For example, such a survey might strive to understand which of the Curriculum 2023 Operating Systems Knowledge Units (or aspects thereof) instructors cover in assigned homework or projects and what students are asked to do in those assignments. Such a survey could also allow us to better understand how well the literature supports instructors' preferred perspectives and approaches as well as to better understand differences in OS education around the world. In addition, it could help us understand how well Curriculum 2023's competency-based curriculum aligns with the resources available in the literature. Such a survey would be a non-trivial undertaking, requiring substantial investment by both the researchers and the instructors who teach OSs.

Another limitation related to the survey is its reach. We advertised the survey to the mailing lists for SIGSCSE and SIGOPS. Our thinking was that SIGSCSE would cover the broader CS education community and that SIGOPS would capture OS researchers who teach OSs but are not active within the CS education community. From a reviewer's comment, it is clear that this approach to data collection

did not capture sufficient responses from faculty who teach OSs. Specifically, that reviewer points out that four of the five faculty who teach OSs at their institution leverage Nachos for the projects, but the survey responses found no one using Nachos. Going forward, this type of research could really use a reliable means of gathering information from faculty who teach a particular course. This could be a valuable service provided by a professional organization, such as SIGOPS or SIGCSE or even the Accreditation Board for Engineering and Technology, but would require strict moderation to limit the amount of traffic in order to maximize the size of the instructor community willing to subscribe to the distribution list.

Finally, a few papers [61, 63] discussed presenting students with principles of systems programming. An area of future work would be to build consensus about a set of principles appropriate for undergraduate system students. It would be interesting to examine how such principles could be incorporated more broadly into an undergraduate course in OSs and perhaps to other courses at the undergraduate level as well.

7 Conclusions

This investigation successfully answered the questions originally posed. Instructors looking to teach OSs for the first time or those wishing to refresh an existing OSs course can use this analysis to understand how other instructors teach OSs. They can also use this analysis to find literature relevant to the approach and perspective they wish to take. They can review the key topics as well as identify the broad set of systems on which other instructors have based projects.

This article also provides a high-level view of the OS educational literature. We uncovered an apparent mismatch between the ways instructors teach OSs and the resources available in the research literature. This mismatch suggests a need for more research in the area of abstract approaches to teaching OSs as well as those in the area of teaching OSs from an external perspective. We also found that educational research in the area of OSs has limited evaluation, typically only anecdotal in nature. Little research has been done to show the efficacy of OS educational tools, systems, and techniques. The papers tend to have few citations, but a respectable number of downloads given the relatively small audience likely to be interested in the results. Our conclusion is that educators appear to be reading these papers, presumably to influence their teaching, but are not actively researching the teaching of OSs, and therefore not citing the work. We also discovered that many individuals and institutions have innovated in the space of OS education, but only two authors (and their institutions) have been prolific.

This literature review also identified areas for future investigation. One such area of investigation is expanding the coverage of the literature review, looking beyond SIGCSE and SIGOPS. Another is a systematic examination of specific assignments/projects for each key topic in Curriculum 2023 [48], both in the literature and in practice among current instructors.

Tanenbaum argues that students of OSs learn best by doing [86]. This position has been widely accepted by the educational community and is one shared by the author. Many of the projects analyzed in this article support an OSs course taught “by doing.” We hope this article will support OS educators as they design or refresh their courses. We also hope that the findings in this investigation will help CS education researchers working to support undergraduate OS education as they choose RQs, build tools, and evaluate them in the classroom.

Acknowledgments

The author gratefully acknowledges the following individuals for responding to the survey used to collect the data reported in this article. Their participation in the study does not imply their endorsement of the study nor their agreement with the conclusions. Participants included: Remzi Arpaci-Dusseau, Antonio Barbalace, João Pedro Barreto, G. Robert Baumgartl, Frank Bellosa, Ken

Birman, Jeremiah Blanchard, Herbert Bos, Lubomír Bulej, Warren R. Carithers, Douglas Comer, Thomas W. Doepfner, David A. Eckhardt, Branden Ghena, Mikey Goldweber, Kyle Hale, Britton Horn, John Hunt, M. F. Kaashoek, Sanidhya Kashyap, Brian Kell, Shiwu Lo, Daniel Lohmann, Barton Miller, Laura Sandoval Montao, Alison N. Norman, Victor Norman, Atsushi Nunome, Sunjae Park, Fernando Pedone, Simon Peter, Jeffrey Pfaffmann, Miroslav Popovic, Joël Porquet-Lupine, Karen Reid, Michael Robbelloth, Zhong Shao, Valerio Schiavoni, Andrew Scott, Ing. Daniel Sol Llaven, Michael Spear, Stu Steiner, John A. Stratton, Michael Swift, Jim Tesesco, Scott Thede, Dwayne Towell, Abhijat Vichare, Gunnar Wolf, Wenxin Zheng, as well as the other respondents who requested to remain anonymous.

A special thanks to the reviewers for their insightful comments and excellent suggestions for improving this article. Thanks to COL Ben Wallen for the inspiration for this literature review and to members of my department for encouragement along the way.

References

- [1] Joel C. Adams and W. David Laverell. 2005. Configuring a Multi-Course Lab for System-Level Projects. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, 525–529. DOI: <https://doi.org/10.1145/1047344.1047509>
- [2] Charles L. Anderson and Minh Nguyen. 2005. A Survey of Contemporary Instructional Operating Systems for Use in Undergraduate Courses. *J. Comput. Sci. Coll.* 21, 1 (Oct. 2005), 183–190. Retrieved from https://www.researchgate.net/publication/228345437_A_survey_of_contemporary_instructional_operating_systems_for_use_in_undergraduate_courses
- [3] Thomas Anderson and Michael Dahlin. 2015. *Operating Systems: Principles & Practice* (2nd ed.). Recursive Books, Ltd. Retrieved from <https://ospp.cs.washington.edu/blog.html>
- [4] Jeremy Andrus and Jason Nieh. 2012. Teaching Operating Systems Using Android. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*, Vol. 43, ACM Press, New York, NY, 613–618. DOI: <https://doi.org/10.1145/2157136.2157312>
- [5] Remzi H. Arpaci-Dusseau. 2018. *OSTEP Projects for an Operating Systems Class*. GitHub. Retrieved June 7, 2023 from <https://github.com/remzi-arpacidusseau/ostep-projects>
- [6] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. 2018. *Operating Systems: Three Easy Pieces* (1.00 ed.). Arpaci-Dusseau Books. Retrieved from <https://pages.cs.wisc.edu/~remzi/OSTEP/>
- [7] Benjamin Atkin and Emin Gün Sirer. 2002. PortOS: An Educational Operating System for the Post-PC Environment. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education ((SIGCSE '02)*. ACM, New York, NY, 116–120. DOI: <https://doi.org/10.1145/563340.563384>
- [8] Adam J. Aviv, Vin Mannino, Thanat Owlarn, Seth Shannin, Kevin Xu, and Boon Thau Loo. 2012. Experiences in Teaching an Educational User-Level Operating Systems Implementation Project. *SIGOPS Oper. Syst. Rev.* 46, 2 (July 2012), 80–86. DOI: <https://doi.org/10.1145/2331576.2331588>
- [9] Maurice Bach. 1986. *The Design of the UNIX Operating System* (1st ed.). Prentice-Hall, USA. Retrieved from <https://archive.org/details/DesignUNIXOperatingSystem>
- [10] Lubomir Bic. 2022. *Operating Systems*. zyBooks, USA. Retrieved from <https://www.zybooks.com/catalog/operating-systems/>
- [11] Michael D. Black. 2009. Build an Operating System from Scratch: A Project for an Introductory Operating Systems Course. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*. ACM, New York, NY, 448–452. DOI: <https://doi.org/10.1145/1508865.1509022>
- [12] Jeremiah Blanchard. 2019. *Reptilian OS Project*. University of Florida. Retrieved July 8, 2023 from <https://www.cise.ufl.edu/research/reptilian/wiki/doku.php>
- [13] Daniel P. Bovet and Marco Cesati. 2001. A Real Bottom-Up Operating Systems Course. *SIGOPS Oper. Syst. Rev.* 35, 1 (Jan. 2001), 48–60. DOI: <https://doi.org/10.1145/371455.371461>
- [14] Randal E. Bryant and David R. O'Hallaron. 2015. *Computer Systems: A Programmer's Perspective* (3rd ed.). Pearson, NY. Retrieved from <https://csapp.cs.cmu.edu/3e/pieces/preface3e.pdf>
- [15] Dennis Brylow. 2008. An Experimental Laboratory Environment for Teaching Embedded Operating Systems. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, New York, NY, 192–196. DOI: <https://doi.org/10.1145/1352135.1352201>

- [16] Steve Carr, Ping Chen, Timothy R. Jozwowski, Jean Mayo, and Ching-Kuang Shene. 2002. Channels, Visualization, and Topology Editor. In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '02)*. ACM, New York, NY, 106–110. DOI: <https://doi.org/10.1145/544414.544448>
- [17] Steve Carr and Ching-Kuang Shene. 2000. A Portable Class Library for Teaching Multithreaded Programming. In *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE conference on Innovation and Technology in Computer Science Education (ITiCSE '00)*. ACM, New York, NY, 124–127. DOI: <https://doi.org/10.1145/343048.343138>
- [18] Haibo Chen and Yubin Xia. 2023. *Modern Operating System: Principle and Implementation*. China Machinery Industry Press, China. Retrieved from <https://ipads.se.sjtu.edu.cn/mospi/>
- [19] Richaerd Chien and Huidong Xu. 2022. *ChCore*. GitHub. Retrieved July 8, 2023 from <https://github.com/WilliamX1/ChCore>
- [20] Wayne A. Christopher, Steven J. Procter, and Thomas E. Anderson. 1993. The Nachos Instructional Operating System. In *USENIX Winter 1993 Conference (USENIX '93)*. USENIX Association, Berkeley, CA. Retrieved from <https://www.usenix.org/legacy/publications/library/proceedings/sd93/christopher.pdf>
- [21] Mark Claypool, David Finkel, and Craig Wills. 2001. An Open Source Laboratory for Operating Systems Projects. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '01)*. ACM, New York, NY, 145–148. DOI: <https://doi.org/10.1145/377435.377669>
- [22] Douglas Comer. 2015. *Operating System Design: The XINU Approach* (2nd ed.). CRC Press, Boca Raton, FL. Retrieved from <https://archive.org/details/operatingsystemd000come>
- [23] Russ Cox, Frans Kaashoek, and Robert Morris. 2020. *xv6: A Simple, Unix-Like Teaching Operating System*. MIT. Retrieved April 26, 2022 from <https://pdos.csail.mit.edu/6.828/2021/xv6/book-riscv-rev2.pdf>
- [24] Michelle Craig and Briana B. Morrison. 2023. *Engage CS Edu*. EngageCSEdu. Retrieved June 7, 2023 from <https://www.engage-csedu.org/>
- [25] Christoffer Dall and Jason Nieh. 2014. Teaching Operating Systems Using Code Review. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, 549–554. DOI: <https://doi.org/10.1145/2538862.2538894>
- [26] Renzo Davoli. 2004. Teaching Operating Systems Administration with User Mode Linux. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '04)*. ACM, New York, NY, 112–116. DOI: <https://doi.org/10.1145/1007996.1008027>
- [27] Harvey M. Deitel, Paul J. Deitel, and David R. Choffnes. 2003. *Operating Systems* (3rd. ed.). Pearson/Prentice Hall, Upper Saddle River, NJ. Retrieved from <https://archive.org/details/operatingsystems0000deit>
- [28] Peter DeRosa, Kai Shen, Christopher Stewart, and Jonathan Pearson. 2006. Realism and Simplicity: Disk Simulation for Instructional OS Performance Evaluation. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '06)*. ACM, New York, NY, 308–312. DOI: <https://doi.org/10.1145/1121341.1121436>
- [29] John Dickinson. 2000. Operating Systems Projects Built on a Simple Hardware Simulator. In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (SIGCSE '00)*. ACM, New York, NY, 320–324. DOI: <https://doi.org/10.1145/330908.331878>
- [30] Thomas W. Doepfner. 2010. *Operating Systems in Depth*. Wiley, India. Retrieved from <https://archive.org/details/operatingsystems0000doep>
- [31] John L. Donaldson. 2001. Architecture-Dependent Operating System Project Sequence. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '01)*. ACM, New York, NY, 322–326. DOI: <https://doi.org/10.1145/364447.364613>
- [32] John L. Donaldson. 2008. Implementation of Threads as an Operating Systems Project. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, New York, NY, 187–191. DOI: <https://doi.org/10.1145/1352135.1352200>
- [33] Zhong Shao, and The Flint Group. 2001. *Certified Kit Operating System*. Yale University. Retrieved July 8, 2023 from <https://flint.cs.yale.edu/certikos/mcertikos.html>
- [34] Bryan Ford, Godmar Back, Greg Benson, Jay Lepreau, Albert Lin, and Olin Shivers. 1997. The Flux OSKit: A Substrate for Kernel and Language Research. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP '97)*. ACM, New York, NY, 38–51. DOI: <https://doi.org/10.1145/268998.266642>
- [35] Michael Goldweber, Renzo Davoli, and Mauro Morsiani. 2005. The Kaya OS Project and the MPS Hardware Emulator. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '05)*. ACM, New York, NY, 49–53. DOI: <https://doi.org/10.1145/1067445.1067462>
- [36] Kyle C. Hale and Peter A. Dinda. 2015. A Case for Transforming Parallel Runtimes Into Operating System Kernels. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15)*. ACM, New York, NY, 27–32. DOI: <https://doi.org/10.1145/2749246.2749264>
- [37] Kyle C. Hale, Peter A. Dinda and HEXSA-Lab. 2014. *Nautilus*. Illinois Institute of Technology. Retrieved August 3, 2023 from <https://github.com/HEXSA-Lab/nautilus>

- [38] Roelof Hamberg and Frits Vaandrager. 2008. Using Model Checkers in an Introductory Course on Operating Systems. *SIGOPS Oper. Syst. Rev.* 42, 6 (Oct. 2008), 101–111. DOI : <https://doi.org/10.1145/1453775.1453793>
- [39] Rob Hess and Paul Paulson. 2010. Linux Kernel Projects for an Undergraduate Operating Systems Course. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*, Vol. 41, ACM Press, New York, NY, 485–489. DOI : <https://doi.org/10.1145/1734263.1734428>
- [40] John M. D. Hill, Clark K. Ray, Jean R. S. Blair, and Curtis A. Carver. 2003. Puzzles and Games: Addressing Different Learning Styles in Teaching Operating Systems Concepts. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)*. ACM, New York, NY, 182–186. DOI : <https://doi.org/10.1145/611892.611964>
- [41] David A. Holland, Ada T. Lim, and Margo I. Seltzer. 2002. A New Instructional Operating System. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '02)*. ACM, New York, NY, 111–115. DOI : <https://doi.org/10.1145/563340.563383>
- [42] David Hovemeyer, Jeffrey K. Hollingsworth, and Bobby Bhattacharjee. 2004. Running on the Bare Metal with GeekOS. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, 315–319. DOI : <https://doi.org/10.1145/971300.971411>
- [43] James Howatt. 2002. Operating Systems Projects: Minix Revisited. *SIGCSE Bull.* 34, 4 (Dec. 2002), 109–111. DOI : <https://doi.org/10.1145/820127.820179>
- [44] Greg Kawell. 2007. Concepts to Real World Implementation via Service Learning. *SIGCSE Bull.* 39, 4 (Dec. 2007), 113–116. DOI : <https://doi.org/10.1145/1345375.1345427>
- [45] The Kernel Development Community. 2015. *FUSE*. The Linux Kernel Organization. Retrieved July 8, 2023 from <https://www.kernel.org/doc/html/next/filesystems/fuse.html>
- [46] Dan Kimmel and J. Rassi. 2018. *Weenix*. Brown University. Retrieved July 8, 2023 from <https://github.com/brown-cs1690/handout/wiki/Weenix-Operating-System>
- [47] Barbara Kitchenham and S. Charters. 2007. *Guidelines for performing Systematic Literature Reviews in software engineering*. EBSE Technical Report EBSE-2007-01. Keele University, Staffordshire, UK. 65 pages. Retrieved from https://www.researchgate.net/publication/258968007_Kitchenham_B_Guidelines_for_performing_Systematic_Literature_Reviews_in_software_engineering_EBSE_Technical_Report_EBSE-2007-01
- [48] Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. 2024. *Computer Science Curricula 2023*. ACM, New York, NY. Retrieved from <https://dl.acm.org/doi/book/10.1145/3664191>
- [49] Oren Laadan, Jason Nieh, and Nicolas Viennot. 2011. Structured Linux Kernel Projects for Teaching Operating Systems Concepts. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*, Vol. 42, ACM, New York, NY, 287–292. DOI : <https://doi.org/10.1145/1953163.1953250>
- [50] Barry Lawson and Lewis Barnett. 2008. Using IPodLinux in an Introductory OS Course. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, New York, NY, 182–186. DOI : <https://doi.org/10.1145/1352135.1352199>
- [51] Haifeng Liu, Xianglan Chen, and Yuchang Gong. 2007. BabyOS: A Fresh Start. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*. ACM, New York, NY, 566–570. DOI : <https://doi.org/10.1145/1227310.1227499>
- [52] Daniel Sol Llaven. 2016. *Sistema operativo panorama para ingeniería en computación e informática*. Grupo Editorial Patria, Mexico City. Retrieved from https://books.google.com/books/about/Sistemas_Operativos_Panorama_para_la_Ing.html?id=WS2QnQAACAAJ
- [53] Luiz Paulo Maia, Francis Berenger Machado, and Ageu C. Pacheco. 2005. A Constructivist Framework for Operating Systems Education: A Pedagogic Proposal Using the SOSim. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '05)*. ACM, New York, NY, 218–222. DOI : <https://doi.org/10.1145/1067445.1067505>
- [54] Jose Alves Marques, Carlos Riberio, Luis Veiga, and Paula Ferreira e Rodrigo Rodrigues. 2011. *Sistemas Operacionais* (2nd. ed.). LTC, Portugal.
- [55] Philipp Mayring. 2000. Qualitative Content Analysis. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research* 1, 2 (June 2000), 10 pages. DOI : <https://doi.org/10.17169/fqs-1.2.1089>
- [56] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. 2019. A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Trans. Educ.* 62, 2 (2019), 77–90. DOI : <https://doi.org/10.1109/TE.2018.2864133>
- [57] Tyrone Nicholas and Jerzy A. Barchanski. 2001. TOS: An Educational Distributed Operating System in Java. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '01)*. ACM, New York, NY, 312–316. DOI : <https://doi.org/10.1145/364447.364611>

- [58] Jason Nieh and Chris Vaill. 2005. Experiences Teaching Operating Systems Using Virtual Platforms and Linux. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, 520–524. DOI: <https://doi.org/10.1145/1047344.1047508>
- [59] Linda Null and Karishma Rao. 2005. CAMERA: Introducing Memory Concepts via Visualization. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, 96–100. DOI: <https://doi.org/10.1145/1047344.1047389>
- [60] Darragh O'Brien. 2017. Teaching Operating Systems Concepts with SystemTap. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, 335–340. DOI: <https://doi.org/10.1145/3059009.3059045>
- [61] Rafael Román Otero and Alex A. Aravind. 2015. MiniOS: An Instructional Platform for Teaching Operating Systems Projects. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. Vol. 46, ACM, New York, NY, 430–435. DOI: <https://doi.org/10.1145/2676723.2677299>
- [62] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering 17* (June 2008), 1–10. Retrieved from <https://www.scienceopen.com/hosted-document?doi=10.14236/ewic/EASE2008.8>
- [63] Ben Pfaff, Anthony Romano, and Godmar Back. 2009. The Pintos Instructional Operating System Kernel. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*. ACM, New York, NY, 453–457. DOI: <https://doi.org/10.1145/1508865.1509023>
- [64] Andrew T. Phillips and Jack S. E. Tan. 2003. Exploring Security Vulnerabilities by Exploiting Buffer Overflow Using the MIPS ISA. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)*. ACM, New York, NY, 172–176. DOI: <https://doi.org/10.1145/611892.611962>
- [65] Andreas Polze and Dave Probert. 2006. Teaching Operating Systems: The Windows Case. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '06)*. ACM, New York, NY, 298–302. DOI: <https://doi.org/10.1145/1121341.1121434>
- [66] Miroslav Popovic, Vladimir Marinkovic, and Vladimir Kovacevic. 2022. *Real Time Operating Systems* (1st. ed.). Novi Sad: Faculty of Technical Sciences, Serbia.
- [67] Marina Prvan and Julije OžEGOVIĆ. 2020. Methods in Teaching Computer Networks: A Literature Review. *ACM Trans. Comput. Educ.* 20, 3, Article 19 (June 2020), 35 pages. DOI: <https://doi.org/10.1145/3394963>
- [68] Kenneth A. Reek. 2002. The Well-Tempered Semaphore: Theme with Variations. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '02)*. ACM, New York, NY, 356–359. DOI: <https://doi.org/10.1145/563340.563477>
- [69] Kenneth A. Reek. 2004. Design Patterns for Semaphores. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, 320–324. DOI: <https://doi.org/10.1145/971300.971412>
- [70] Steven Robbins. 2000. Experimentation with Bounded Buffer Synchronization. In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (SIGCSE '00)*. ACM, New York, NY, 330–334. DOI: <https://doi.org/10.1145/330908.331880>
- [71] Steven Robbins. 2001. Starving Philosophers: Experimentation with Monitor Synchronization. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '01)*. ACM, New York, NY, 317–321. DOI: <https://doi.org/10.1145/364447.364612>
- [72] Steven Robbins. 2003. Using Remote Logging for Teaching Concurrency. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)*. ACM, New York, NY, 177–181. DOI: <https://doi.org/10.1145/611892.611963>
- [73] Steven Robbins. 2004. A Disk Head Scheduling Simulator. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, 325–329. DOI: <https://doi.org/10.1145/971300.971413>
- [74] Steven Robbins. 2005. An Address Translation Simulator. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, 515–519. DOI: <https://doi.org/10.1145/1047344.1047507>
- [75] Steven Robbins. 2006. A UNIX Concurrent I/O Simulator. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '06)*. ACM, New York, NY, 303–307. DOI: <https://doi.org/10.1145/1121341.1121435>
- [76] Steven Robbins. 2008. A Three Pronged Approach to Teaching Undergraduate Operating Systems. *SIGOPS Oper. Syst. Rev.* 42, 6 (Oct. 2008), 93–100. DOI: <https://doi.org/10.1145/1453775.1453792>
- [77] Open Robotics. 2010. *Robot Operating System*. Open Robotics. Retrieved July 8, 2023 from <https://www.ros.org/>
- [78] Alexander Schmidt, Andreas Polze, and Dave Probert. 2010. Teaching Operating Systems: Windows Kernel Projects. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, 490–494. DOI: <https://doi.org/10.1145/1734263.1734429>
- [79] Sadia Sharmin. 2021. Creativity in CS1: A Literature Review. *ACM Trans. Comput. Educ.* 22, 2, Article 16 (Nov. 2021), 26 pages. DOI: <https://doi.org/10.1145/3459995>

- [80] Robert J. Sheehan. 2007. Teaching Operating Systems with Ruby. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '07)*. ACM, New York, NY, 38–42. DOI : <https://doi.org/10.1145/1268784.1268798>
- [81] Kiyoshi Shibayama. 2007. *Operating Systems Learned in Computer Science – OS Science*. Modert Science, Inc., Tokyo.
- [82] Avi Silberschatz, Peter Baer Galvin, and Greg Gagne. 2018. *Operating Systems Concepts* (10th. ed.). John Wiley & Sons, Inc.. Retrieved from <https://archive.org/details/operatingsystemconcepts10th>
- [83] Thomas Spink. 2018. *Informatics Research Operating System*. GitHub. Retrieved July 8, 2023 from <https://github.com/tspink/infos>
- [84] William Stallings. 2011. *Operating Systems: Internals and Design Principles* (9th. ed.). Prentice Hall Press. Retrieved from <https://archive.org/details/operatingsystems0007stal>
- [85] Valdemar Švábenský, Jan Vykopal, and Pavel Čeleda. 2020. What Are Cybersecurity Education Papers About? A Systematic Literature Review of SIGCSE and ITiCSE Conferences. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. ACM, New York, NY, 2–8. DOI : <https://doi.org/10.1145/3328778.3366816>
- [86] Andrew Tanenbaum. 1987. A Unix Clone with Source Code for Operating Systems Courses. *Oper. Syst. Rev.* 21, 1 (Jan. 1987), 10 pages. DOI : <https://doi.org/10.1145/24592.24596>
- [87] Andrew S. Tanenbaum and Herbert Bos. 2015. *Modern Operating Systems* (4th. ed.). Pearson, NY. Retrieved from <https://archive.org/details/modernoperatings0000tane>
- [88] Andrew S. Tanenbaum and Albert S. Woodhull. 2006. *Operating Systems: Design and Implementation* (3rd. ed.). Pearson Prentice Hall, NJ. Retrieved from <https://archive.org/details/operatingsystems00tane>
- [89] Gunnar Wolf Iszaevich, Esteban Ruiz, Federico Bergero, and Erwin Meza. 2015. *Fundamentos de Sistemas Operativos*. Universidad Nacional Autonoma de Mexico, Mexico. Retrieved from <https://ru.iiec.unam.mx/2718/>
- [90] Geoff Wong, Trish Greenhalgh, Gill Westhorp, Jeanette Buckingham, and Ray Pawson. 2013. RAMESES Publication Standards: Meta-Narrative Reviews. *BMC Med.* 11, 20 (2013), 15 pages. DOI : <https://doi.org/10.1186/1741-7015-11-20>
- [91] Michael Ziwiwsky, Kyle Persohn, and Dennis Brylow. 2013. A Down-to-Earth Educational Operating System for Up-in-the-Cloud Many-Core Architectures. *ACM Trans. Comput. Educ.* 13, 1, Article 4 (Feb. 2013), 12 pages. DOI : <https://doi.org/10.1145/2414446.2414450>
- [92] ETH Zürich. 2008. *The Barrelfish Operating System*. ETH Zürich. Retrieved July 8, 2023 from <https://barrelfish.org/>

Appendix

A Survey Questions

Undergraduate OSs

The data from this form will be used as part of a research study examining how OS is taught at the undergraduate level. The data you provide will be combined with data from numerous other respondents around the world.

E-mail: <text entry field>

I understand that my data will be used as part of a research study.

I understand that the results of this study will be reported in an aggregate format with no identifying information.

I understand that my identity and that of my institution are required for my data to be included in the study, but that my identity and that of my institution will be removed from the study data after duplications and conflicting answers have been removed.

- I agree to participate in this research study. → Go to next question
 - I do not wish to participate in this research study. → Thank them and end the survey
-

I am willing to be contacted in the event that multiple people from my institution respond to this survey and provide conflicting data. I understand that if I do not agree to be contacted, my

institution's data may be removed from the data in the event that multiple people submit conflicting information.

- I am willing to be contacted in the event conflicting data are submitted from my institution.
 - I do NOT wish to be contacted in the event conflicting data are submitted from my institution.
-

I would like to be acknowledged in any publications that use this data. The acknowledgment will state: “The authors gratefully acknowledge the following individuals for responding to the survey used to collect the data reported in this article. Their participation in the study does not imply their endorsement of the study nor their agreement with the conclusions. Participants included: <list of names>”

- Yes, please list my name in the acknowledgments.
 - No, please do NOT list my name in the acknowledgments. → Skip next question
-

My name (for acknowledgement purposes) is: <text entry field>

Have you taught operating systems to undergraduates in past the 2 academic years?

- Yes
 - No
-

The name of my institution is: <text entry field>

My institution:

- Public
 - Private
 - Other: <text entry field>
-

What type of institution is this?

- University
 - College
 - Community College
 - Polytechnic
 - Vocational/Trade School/Institute of Technology³
 - CS Boot Camp
 - Distance Learning (primarily)
 - Other: <text entry field>
-

Students taking operating systems at my institution are **primarily**:

- Frosh
 - Sophomores
 - Juniors
 - Seniors
 - Graduate students
 - Other: <text entry field>
-

³For follow-on surveys, we recommend changing the way Institutes of Technology are handled. Some are more like trade schools and others are more like Universities.

Which textbook(s) has your institution used in the past 2 years?

- Operating Systems: Principles and Practice by Thomas Anderson and Michael Dahlin
- Operating Systems: Three Easy Pieces by Remzi Arpaci-Dusseau and Andre Arpaci-Douseau
- The Design of the UNIX Operating System by Maurice Bach
- Operating System Concepts, by Abraham Silberschatz, Greg Gagne, and Peter Galvin [sic]
- Operating System Concepts, by William Stallings
- Modern Operating Systems, by Andrew Tanenbaum and Herbert Bos
- Operating System Design and Implementation by Andrew Tanenbaum and Albert Woodhull
- I don't use a textbook
- Other: <text entry field>

Select any of the following platforms that your institution uses as the basis for one or more projects or assignments in the past 2 years.

- BabyOS
- Embedded Xinu
- FOSL
- GeekOS
- iPodLinux
- Kaya
- Linux
- MiniOS
- Minux
- Nachos
- OS/161
- OSTEP Projects in GitHub
- PennOS
- Pintos
- TOS
- xv6
- Other: <text entry field>

Operating systems can be taught from either an internal or external perspective. An internal perspective focuses on the implementation of an operating system (such as implementing new system calls), whereas an external perspective focuses on using the services of an operating system (such as using system calls). Using the scale below, tell us which perspective you take in your operating systems course.

- Entirely internal
- Mostly internal
- Equally internal and external
- Mostly external
- Entirely external

Operating systems can be taught with either a concrete or abstract approach. A concrete approach is one that focuses on the design and implementation of a realistic operating system. An abstract approach is one that focuses on algorithms and techniques in isolation or in a simulated environment. Using the scale below, tell us which approach you take in your operating systems course.

- Entirely concrete
- Mostly concrete
- Equally concrete and abstract
- Mostly abstract
- Entirely abstract

Thank you for your participation!

Received 11 July 2023; revised 17 July 2024; accepted 1 August 2024