Embedding In-Browser Linux Labs as Smart Content for Intelligent Textbooks

Andrzej Wójtowicz[1]

[1]Adam Mickiewicz University, Faculty of Mathematics and Computer Science, Uniwersytetu Poznańskiego 4, 61-614 Poznań, Poland

Abstract

This paper presents a browser-based Linux laboratory component for intelligent HTML textbooks. The demonstrator embeds a bootable Linux environment alongside the textbook content and runs it locally in the learner's browser using static web assets. The paper focuses on the textbook-facing experience: a short lesson excerpt, several activity variants covering a single topic, a compact system architecture, and a walkthrough of formative assessment within the virtual machine. The technical design separates a reusable base Linux image from small, activity-specific laboratory disks containing files, datasets, setup scripts, and assessment checkers. Optional language-model support is treated as an extension for generating supplementary practice activities. The demonstrator shows that executable operating-system activities can be packaged as smart textbook content without requiring a hosted laboratory service.

Keywords

Linux, emulation, v86, intelligent textbooks, smart content

1. Introduction

Competence with Linux and Unix-like command-line environments is part of contemporary computer science and data science education [1, 2]. However, learners often reach later programming, systems, or data-oriented courses without enough confidence in shell-based work, and limited early exposure can become visible only when more advanced coursework assumes command-line fluency [3].

Introductory Linux learning requires practice in a live operating-system environment. Static online tutorials remain useful [4], but they cannot by themselves support exploration, failure, recovery, and automated checking. Local virtual machines (VMs) reduce hosting costs but can be difficult for beginners to set up. Hosted environments and certification-oriented lab platforms reduce setup work for learners but shift the computational and operational burden to providers [5]. Recent browser-accessible Linux learning systems show the value of lightweight labs [6], yet the lab is still often treated as a site or service separate from the textbook.

This paper reframes a Linux lab as smart content embedded in a textbook: an executable learning object that extends static reading, in line with prior work on intelligent textbooks [7, 8, 9]. Related systems such as Runestone, Jupyter notebooks with nbgrader, and CodeRunner integrate executable programming activities and assessment [10, 11, 12]. Browser-based Linux systems and WebAssembly-based execution environments are also established [13, 14, 15, 16, 17]. The contribution of this demonstrator is the connection between these two lines of work: a bootable Linux environment packaged as modular, inspectable textbook content.

2. Demo Scenario: Linux Activities in an Intelligent Textbook

The demonstrator targets a textbook used in an introductory Linux course for first-semester computer science undergraduates. The following subsections present the lesson context and activities.

iTextbooks'26: Seventh Workshop on Intelligent Textbooks, June 28, 2026, Seoul, Republic of Korea

*Corresponding author.

andre@amu.edu.pl (A. Wójtowicz)

0000-0003-1385-6572 (A. Wójtowicz)

2.1. Textbook Context

The example lesson is placed in a module on command-line data processing and introduces pipelines over small log files. It frames pipelines as a way to combine simple commands such as `grep`, `cut`, `sort`, and `uniq -c` into inspectable data transformations. The learning objective is for learners to filter records, extract fields, count repeated values, and save the result in a checked output file. After reading the short explanation, the learner launches an embedded Linux environment configured for the activity.

2.2. Example Activities

The demonstrator uses several activities for the same topic, showing how the material can be explored in different ways and at different levels of depth. Table 1 shows the intended activities. The first task is author-defined and can be used as a canonical activity. The remaining variants illustrate supplementary practice that can be authored manually or generated by a language model with the author's review.

Table 1
Example activity set for the pipelines lesson.

Activity	Learner task	Expected outcome
1. WARN by service	Count how many WARN log entries belong to each service.	Sorted report file with counts.
2. Failed login users	Extract unique users from failed authentication lines.	One username per line, sorted.
3. Top endpoints	Count requested endpoints with 5xx status codes.	Frequency table of paths.

3. Architecture

The high-level architecture is presented in Figure 1. The demonstrator can be deployed as ordinary static web content: HTML, JavaScript, WebAssembly, a kernel/initrd pair, and activity disk images. The Linux environment boots inside the browser using v86 [18], with a small Buildroot-based Linux image [19]. Xterm.js provides the visible terminal used by the learner [20]. This keeps the textbook deployable on static hosting while moving execution to the learner's device.

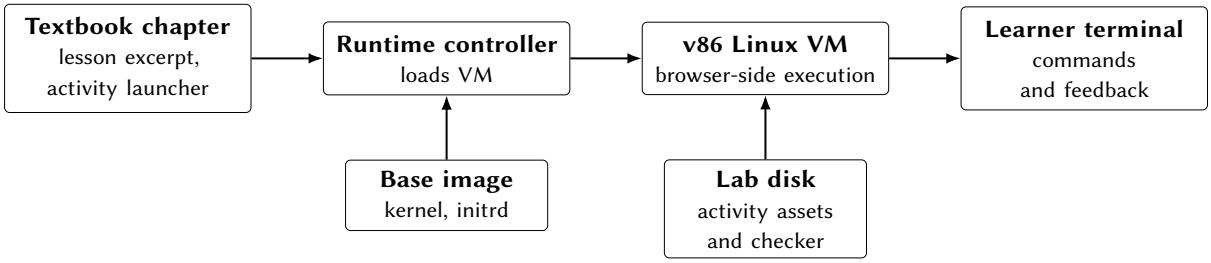

Figure 1: High-level architecture of the executable textbook component.

3.1. Base Image and Laboratory Disks

The executable content is separated into a module-level base image and activity-level lab disks. The *base image* is the reusable Linux system shared by activities in a larger textbook unit, including the kernel, core tools, accounts, and services. A *lab disk* is a small attachable disk image that adds the exercise-specific files, setup steps, and checks for one task. This avoids rebuilding the full Linux image for every activity while still allowing each textbook section to present a different filesystem state.

3.2. Feedback and Checking

The learner interacts with the main terminal. Setup and checking can be executed through a separate control channel inside the virtual machine. In the demo, checkers inspect the current filesystem state and return formative feedback to the textbook page. Because both execution and checking occur on the learner-controlled client, these checks are intended for practice and self-regulation. Summative grading would require a trusted validation channel, such as server-side verification or integration with a learning management system (LMS).

4. Demonstration Walkthrough

In the scenario discussed here, the learner opens the pipelines lesson, starts the Linux VM for the first practice activity, and receives a task-specific workspace. The learner reads the task description, inspects the input log file, constructs a shell pipeline, saves the requested report, and presses a check button in the textbook interface. The checker runs inside the VM and reports whether the expected file content has been produced. The snapshot is shown in Figure 2.

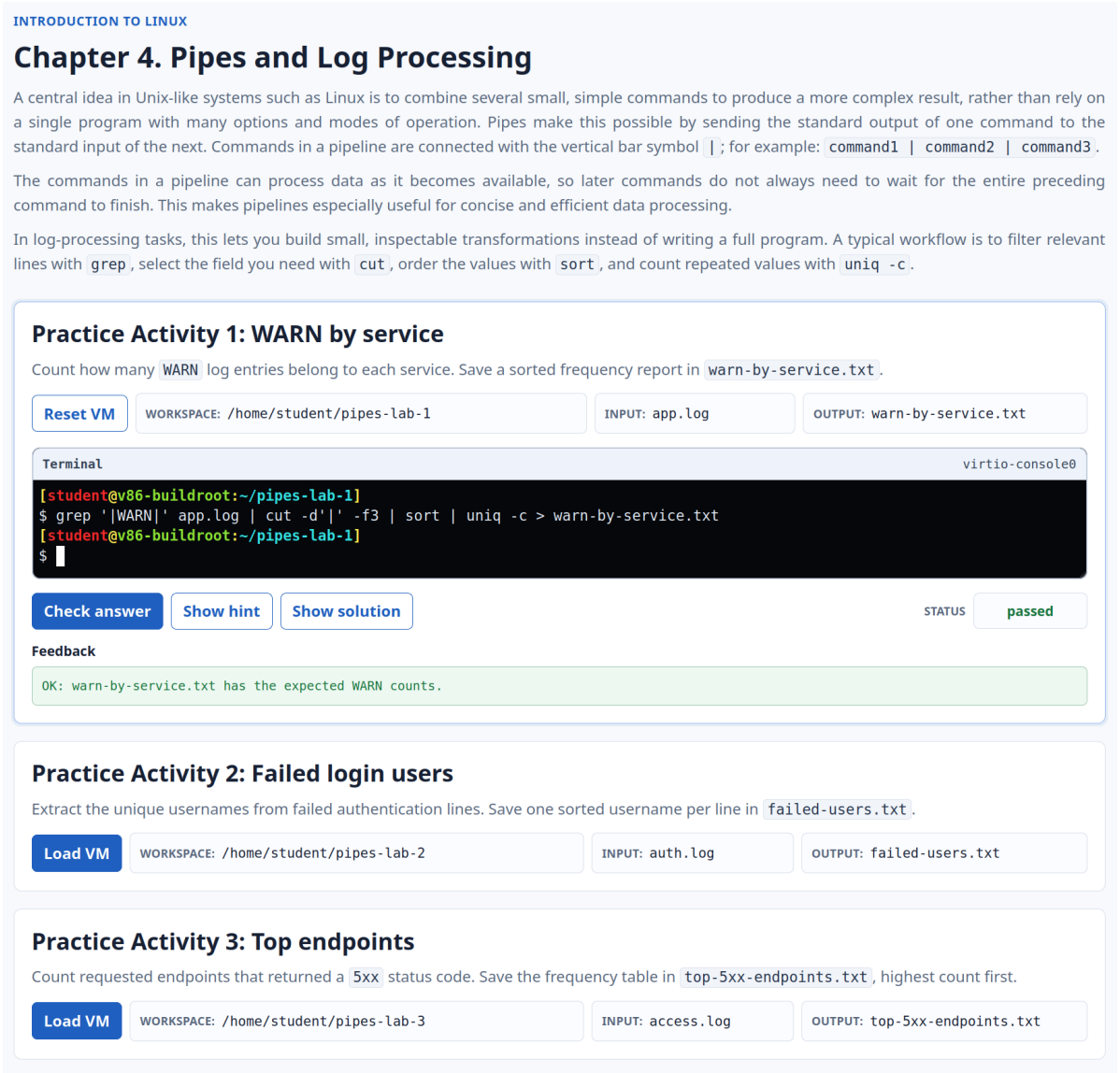

Figure 2: Snapshot of the executable pipelines lesson with the first practice activity loaded. The interface shows the textbook excerpt, task-specific workspace metadata, the learner's shell pipeline in the in-browser Linux terminal, and the checker status with formative feedback.

4.1. Extending the Demo with Generated Practice

The same architecture can support generated practice. A practical deployment path is to let authors generate additional activities in advance, inspect them, and distribute them as ordinary lab disks. However, intelligent textbook support should also allow contextual practice to be customized at use time, e.g., when a learner asks for another exercise related to the current section.

Because the demonstrator is designed to move computation from the server to the learner's device, local generation should be considered first. In this setting, a language model would generate only a compact activity specification: a task title, a learner-facing description, a setup script, and a checker script. Prior work has explored automatic exercise generation and scaffolding with large language models (LLMs) [21, 22, 23]. Browser-side execution is possible in principle using WebGPU and WebLLM-compatible models [24, 25].

Preliminary tests with an off-the-shelf small model, Qwen2.5-Coder-0.5B-Instruct [26], did not produce results reliable enough for unsupervised deployment. For now, high-quality generated practice requires either an external or institutionally hosted LLM API, which weakens the fully local model, or further work on prompt design, validation, and adapting small language models to the textbook's activity format.

5. Feasibility and Limitations

Preliminary checks suggest that the demonstrator is practical as static textbook content. A Buildroot base image used in testing is approximately 38 MB and boots in under 25 seconds on an AMD Ryzen AI 7 350 CPU. A separate optimization experiment suggests that the base image can be reduced to approximately 13 MB, with observed boot time decreasing to about 6 seconds on the same hardware. Activity lab disks are much smaller, typically under 100 KB for text-based datasets and scripts. This makes static hosting realistic; for example, GitHub Pages documents site and bandwidth limits that are compatible with small classroom-scale deployments when browser caching is used [27].

The current work remains a demo, not a classroom evaluation. It does not yet measure learning outcomes, learner persistence, or long-term use. Performance depends on the device and browser, especially if local LLM activity generation is enabled. Client-side checking is suitable for formative feedback but not secure summative assessment. Generated shell scripts also require stronger validation before broad deployment.

6. Conclusion

This paper presents an executable Linux lab as modular smart content for intelligent textbooks. The main idea is to keep the textbook statically deployable while embedding a real, browser-based operating system environment alongside the lesson content. By separating a reusable base Linux image from small laboratory disks and local checkers, the approach supports efficient, context-aware hands-on practice without a hosted lab backend. Optional generated practice can extend the activity set, but further work is needed on validation, local model reliability, and classroom evaluation before such customization can be used broadly.

Declaration on Generative AI

During the preparation of this work, the author used Grammarly and Codex for grammar checks, paraphrasing, rewording, and editing support. The author reviewed and edited the content and takes full responsibility for the publication's content.

References

[1] A. N. Kumar, R. K. Raj, S. G. Aly, M. D. Anderson, B. A. Becker, R. L. Blumenthal, E. Eaton, S. L. Epstein, M. Goldweber, P. Jalote, D. Lea, M. Oudshoorn, M. Pias, S. Reiser, C. Servin, R. Simha, T. Winters, Q. Xiang, Computer Science Curricula 2023, Association for Computing Machinery, New York, NY, USA, 2024. doi:10.1145/3664191.

[2] ACM Data Science Task Force, Computing Competencies for Undergraduate Data Science Curricula, Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3453538.

[3] E. Dillon, K. L. Williams, A. S. Pryor, T. Wimberly Jr., M. McMichael, A. M. Arowolaju, D. B. Davis, T. Ayodele, Exploring the Impact of Exposing Command Line Programming to Early CS Majors (An HBCU Case Study), in: 2024 ASEE Annual Conference & Exposition, ASEE Conferences, Portland, Oregon, 2024, pp. 1–18. doi:10.18260/1-2--47434.

[4] Linux Journey, Tutorials, https://linuxjourney.org/tutorials, 2026. Accessed: 2026-05-10.

[5] Red Hat, Training and Certification, https://www.redhat.com/en/services/training-and-certification, 2026. Accessed: 2026-05-10.

[6] Y. Shoshitaishvili, A. Doupe, C. Nelson, The Linux Luminarium: Learning Linux by Leveraging Lightweight Labs and Ludicrous Lessons, in: Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.1, SIGCSE TS 2026, Association for Computing Machinery, New York, NY, USA, 2026, p. 985–991. doi:10.1145/3770762.3772655.

[7] S. Sosnovsky, P. Brusilovsky, A. Lan, Intelligent Textbooks, International Journal of Artificial Intelligence in Education 35 (2025) 967–986. doi:10.1007/s40593-024-00451-9.

[8] I. Alpizar-Chacon, J. Barria-Pineda, K. Akhuseyinoglu, S. Sosnovsky, P. Brusilovsky, Integrating Textbooks with Smart Interactive Content for Learning Programming, in: Proceedings of the Third International Workshop on Intelligent Textbooks 2021, volume 2895 of *CEUR Workshop Proceedings*, CEUR-WS.org, Online, 2021, pp. 4–18. URL: https://ceur-ws.org/Vol-2895/paper11.pdf.

[9] J. Barria-Pineda, A. B. L. Narayanan, P. Brusilovsky, Augmenting Digital Textbooks with Reusable Smart Learning Content: Solutions and Challenges, in: Proceedings of the Fourth International Workshop on Intelligent Textbooks 2022, volume 3192 of *CEUR Workshop Proceedings*, CEUR-WS.org, Durham, UK, 2022, pp. 77–91. URL: https://ceur-ws.org/Vol-3192/itb22_p8_full9434.pdf.

[10] B. J. Ericson, B. N. Miller, Runestone: A Platform for Free, On-line, and Interactive Ebooks, in: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1012–1018. doi:10.1145/3328778.3366950.

[11] Project Jupyter, D. Blank, D. Bourgin, A. Brown, M. Bussonnier, J. Frederic, B. Granger, T. Griffiths, J. Hamrick, K. Kelley, M. Pacer, L. Page, F. Pérez, B. Ragan-Kelley, J. Suchow, C. Willing, nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook, Journal of Open Source Education 2 (2019) 32. doi:10.21105/jose.00032.

[12] R. Lobb, J. Harlow, Coderunner: A Tool for Assessing Computer Programming Skills, ACM Inroads 7 (2016) 47–51. doi:10.1145/2810041.

[13] F. Bellard, JSLinux, https://bellard.org/jslinux/, 2026. Accessed: 2026-05-10.

[14] Copy, v86 open source project, https://copy.sh/v86/, 2026. Accessed: 2026-05-10.

[15] Leaning Technologies, CheerpX, https://cheerpx.io/, 2026. Accessed: 2026-05-10.

[16] R. Sharrock, L. Angrave, E. Hamonic, WebLinux: a scalable in-browser and client-side Linux and IDE, in: Proceedings of the Fifth Annual ACM Conference on Learning at Scale, L@S '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1–2. doi:10.1145/3231644.3231703.

[17] E. Wen, S. Ma, P. Denny, E. Tempero, G. Weber, Z. Yue, KernelVM: Teaching Linux Kernel Programming through a Browser-Based Virtual Machine, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, SIGCSETS 2025, Association for Computing Machinery, New York, NY, USA, 2025, p. 1204–1210. doi:10.1145/3641554.3701831.

[18] Copy, v86: x86 PC emulator and x86-to-wasm JIT, running in the browser, https://github.com/copy/v86, 2026. Accessed: 2026-05-10.

[19] Buildroot, Buildroot open source project, https://buildroot.org/, 2026. Accessed: 2026-05-10.
[20] Xterm.js, Xterm.js open source project, https://xtermjs.org/, 2026. Accessed: 2026-05-10.
[21] W. Aerts, G. Fletcher, D. Miedema, A Feasibility Study on Automated SQL Exercise Generation with ChatGPT-3.5, in: Proceedings of the 3rd International Workshop on Data Systems Education: Bridging Education Practice with Education Research, DataEd '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 13–19. doi:10.1145/3663649.3664368.
[22] R. Van Campenhout, M. Clark, B. Jerome, J. S. Dittel, B. G. Johnson, Advancing Intelligent Textbooks with Automatically Generated Practice: A Large-Scale Analysis of Student Data, in: Proceedings of the Fifth International Workshop on Intelligent Textbooks 2023, volume 3444 of *CEUR Workshop Proceedings*, CEUR-WS.org, Tokyo, Japan, 2023, pp. 15–26. URL: https://ceur-ws.org/Vol-3444/itb23_s1p2.pdf.
[23] A. Durg, C. Kultur, A. Zhang, J. Savelka, LLM-powered Framework for Automatic Generation of Metacognitive Scaffolding Cues for Introductory Programming in Higher Education, in: Proceedings of the Sixth International Workshop on Intelligent Textbooks 2025, volume 4010 of *CEUR Workshop Proceedings*, CEUR-WS.org, Palermo, Italy, 2025, pp. 25–36. URL: https://ceur-ws.org/Vol-4010/itb25_s1p3.pdf.
[24] W3C, WebGPU, https://webgpu.org/, 2026. Accessed: 2026-05-10.
[25] Machine Learning Compilation community, WebLLM, https://webllm.mlc.ai/, 2026. Accessed: 2026-05-10.
[26] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu, K. Dang, Y. Fan, Y. Zhang, A. Yang, R. Men, F. Huang, B. Zheng, Y. Miao, S. Quan, Y. Feng, X. Ren, X. Ren, J. Zhou, J. Lin, Qwen2.5-Coder Technical Report, 2024. arXiv:2409.12186.
[27] GitHub Pages, Limits, https://docs.github.com/en/pages/getting-started-with-github-pages/github-pages-limits, 2026. Accessed: 2026-05-10.