ShellOnYou : Learning by Doing Unix Command Line

Vincent Berry, Arnaud Castelltort, Chrysta Pélissier, Marion Rousseau, Chouki Tibermacine

ShellOnYou: Learning by Doing Unix Command Line

Vincent Berry*
LIRMM - Univ Montpellier, CNRS
Montpellier, France
vincent.berry@umontpellier.fr

Arnaud Castelltort
LIRMM - Univ Montpellier, CNRS
Montpellier, France
Arnaud.Castelltort@umontpellier.fr

Chrysta Pelissier
LHUMAIN - Univ Paul Valéry
Montpellier 3
Montpellier, France
chrysta.pelissier@umontpellier.fr

Marion Rousseau
Fondation Polytech
Nantes, France
marion.rousseau@polytech-reseau.org

Chouki Tibermacine
LIRMM - Univ Montpellier, CNRS
Montpellier, France
Chouki.Tibermacine@umontpellier.fr

ABSTRACT

We present ShellOnYou, a new Computer Science education tool, and analyze its use with four successive student cohorts.

Developed to help instructors manage numerous students, this web application offers auto-graded exercises to acquire practical knowledge of Unix-like operating systems from the command line. For each answer, and almost instantly, students receive a score and detailed feedback. This reactive and iterative process encourages students to resubmit answers and progressively expand their procedural knowledge. The tool can also deliver individualized statements, thereby allowing students to improve their skills by combining personal research and peer learning. As an online tool, ShellOnYou affords students access flexibility, and also easily fits in distance learning programs. We found it particularly useful when teaching students with heterogeneous Unix backgrounds. The tool is available on request.

We placed four successive student cohorts in a learning situation involving this tool, and asked them to fill a survey at the end of the learning period. We combine qualitative and quantitative methods to analyze their answers to the survey. We attempt to characterize their acquisition of procedural knowledge and the building of group dynamics. Several dimensions emerge: the benefits of using the tool in a learning situation, the learning process iteration as a catalyst for renewed commitment, the tool's entertaining format and its scoring system as a motivation for regular studying, and the inherent customization of the learning pace.

CCS CONCEPTS

• **Software and its engineering** → **Operating systems**; **Operating systems**; • **Social and professional topics** → **Computing education**; **Computer science education**; **Computing education**; **Computer science education**; • **Applied computing** → **Distance learning**; **Distance learning**.

KEYWORDS

learning by doing, practical knowledge, Unix command line interface, autograder, automated feedback, students' perception.

ACM Reference Format:
Vincent Berry, Arnaud Castelltort, Chrysta Pelissier, Marion Rousseau, and Chouki Tibermacine. 2022. ShellOnYou: Learning by Doing Unix Command Line. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3502718.3524753

1 INTRODUCTION

Unix introduction courses aim at providing students with conceptual and practical knowledge to efficiently interact with this ubiquitous system, and in particular to do it from the command line interface (CLI) [23]. These courses feature in the first year of many Computer Science (CS) programs. Due to the predominance of the Unix system in computing facilities, students in many other fields (mathematics, physics, natural and social sciences, *e.g.*, see [17]) may also be interested in Unix introduction courses. As a result, many universities end up having to tutor large cohorts of students who need to acquire basic Unix practical knowledge, and more particularly to learn how to manage Unix from the CLI.

In our case, we prepare students for CS careers within a 3-year program leading to an engineering degree. We teach an Operating System (OS) course at the very beginning of the students' first term in our curriculum. This course introduces them to the basics of a Unix system, and specifically the CLI. Indeed, as in other CS curricula, knowing one's way around Unix is a prerequisite to many other courses; students with a deficit on this topic might find it difficult to graduate. Generally, the students who enroll in our curriculum have various backgrounds: some have never heard of Unix systems; most have used Unix in strictly minimal ways within a programming course in a multidisciplinary curriculum; others have used it on a regular basis in a pure CS undergraduate curriculum. This echoes the observations of Moy [20] in a similar context. Moreover, upon starting their first semester, very few students know each other: indeed, they originate from various geographical locations and

ITiCSE 2022, July 8–13, 2022, Dublin, Ireland.

ACM ISBN 978-1-4503-9201-3/22/07…$15.00
https://doi.org/10.1145/3502718.3524753

qualify for our CS program via a national examination procedure. For various institutional reasons, we have a relatively small number of instruction hours to dedicate to this topic. On the other hand, we manage cohorts of only 40 to 50 students, while other programs commonly require that introductory OS courses be delivered to several hundred students simultaneously. In the latter case, one of the difficulties is to find enough qualified instructors. This becomes an acute issue in computer literacy courses [13] where survival skills in Unix are required from all first-year students entering a program.

To actively promote the acquisition of practical knowledge of the Unix CLI, we needed to create a dedicated learning situation. Among others, the hands-on learning approach most often yields actionable feedback from students and is advocated as an efficient and engaging approach for students. [19, 21, 25]. Accordingly, we wanted practical activities to be at the core of the learning situation we would design. However, this approach tends to generate a considerable volume of student work to be examined and graded by instructors (*e.g.*, [14]). Since we were constrained by the number of class hours (i.e. face time with students), we wanted to design a tool which would automate the examination process. Though the Unix CLI can be considered from a programming perspective (shell scripts), we believe it can only be effectively grasped by using it interactively within a Unix environment. Following Doane et al. [9], we wanted to *measure the students' performance in tasks requiring them to comprehend and produce Unix commands*. For this reason, we ruled out assessment websites or learning management system (LMS) plugins centered on programming skills. [11, 16]. Additionally, with only a relatively small number of teaching hours being funded, we could not commit to online paid plans: we thus ruled out e-recruitment platforms [1–3] and publisher tools [4, 5].

We therefore decided to develop a custom web-based tool; specifically, we wanted it to provide automated feedback to students. This tool, called ShellOnYou, is free and can be easily installed as an autonomous web application thanks to Docker technology. An instance is currently running at https://www.shellonyou.fr. The exercises proposed in the tool can also be integrated as activities within an LMS to provide a seamless experience. Due to article length constraints, below we provide only a survey of the tool's functionalities (Section 2). A subsequent paper will present the tool's in-depth features and guidelines for setting up exercise sessions.

Applying this tool to a learning situation allowed us to meet a two-fold objective: *i)* to increase our students' skillset *i.e.*, *procedural* [18] knowledge (hard skills) of the Unix CLI, but also *ii)* to foster team dynamics that would benefit the entire cohort in the subsequent learning phases of the curriculum (in our case, students share 80% of their courses during the 3-year program). The learning situation is described below in Section 3.1.

We set up this learning situation for four successive student cohorts in the Fall term, from 2018 to 2021. We assessed it by asking students to fill surveys and participate to interviews. Our main focus question was: "*How do this tool and this learning situation favor the acquisition of practical knowledge in computer science?*". Section 4 reports our findings when analyzing students' answers.

2 THE SHELLONYOU TOOL

2.1 Principles of the exercise platform

Instructors propose *exercises*, composed of a template (*i.e.*, generic) statement (plain text or HTML format), a template archive (*i.e.*, a tarball), and two Python scripts; these two artifacts effectively provide code to randomly individualize student statements, as well as code to analyze their answers. This template approach allows instructors to give slightly different versions of the same exercise to different students, so as to prevent basic plagiarism. Indeed, within this framework, students find that an expedient copy/paste approach is unrewarding, and therefore unproductive. Our goal, however, is not to impede communication between students; instead, we want to ensure that student interactions lead to the full comprehension of instructions and the active sharing of useful tips to solve the exercises; mainly, we just intend to provide ways to limit the sharing of complete solutions. In our experiments, we encouraged students to share their thoughts and work collaboratively; we went to the extent of booking a lab room for them so they could work in close teams, and referring them to social network platforms for further communications. This helped promote knowledge transfer between peers and the development of group dynamics.

Students access exercises on the ShellOnYou web platform by enrolling into an exercise *session* set up by an instructor. Sessions are available for a specific time period and can be set up to require an access password. Each session consists in an ordered list of available exercises: students can select and go through the exercises at their discretion, or they can be constrained to access an exercise only after achieving a minimum score on the previous one.

For a typical exercise, students must first download a tar archive consisting of a small file structure, and explore this files using the instructions given on the exercise page. The instructions require students to perform several tasks within the file structure, or to complete the answer file it contains. Students then submit their work by creating an archive of the modified file structure and uploading it onto the ShellOnYou platform. The platform then provides them with detailed feedback on the various items mentionned in the exercise statement, and outputs an overall performance score. This feedback offers a list of wrong answers and misunderstood items. Thus, students can learn from their mistakes, return to their work and submit another, improved answer. This iterative learning process can continue until the exercise becomes unavailable (i.e. when the deadline for the exercise or for the session has been met). Personalized feedback allows students to improve their performance gradually, and to ultimately master the desired skillset. Obviously, this objective is fully achieved when the exercise level is aligned with the tutoring experience delivered outside the platform [6]. Based on our observations, receiving a score upon each attempt and being allowed further attempts combine to generate an engagement factor for students, who seem to get caught into a game-like spiral. This factor is positively reinforced when they exchange scores with classmates in the room or via social networks.

The feedback given to students is obtained by running the code proposed by the creator of the exercise; this code behaves like a series of unit tests, such as those commonly used in the software development industry (as in the *Codeboard* tool [11]). Generally, for each specific exercise, instructors know the answer pitfalls they

have to design against, so their students learn to avoid them. Building this knowledge into the very code which analyses the answers allows instructors to provide their expertise to hundreds of students simultaneously and ubiquitously, i.e. anytime and anywhere students find it convenient to access the platform. Indeed, our experiments show that students work from various places and at greatly varying times, including evenings and nights.

For each session they organize, instructors know which student submitted their answer and when it was submitted; they can also access the resulting feedback and scores for each student. More details on how to use the tool - as a student or as an instructor - are provided in the help pages.

2.2 Advantages and drawbacks of the ShellOnYou tool

ShellOnYou offers an iterative learning process in which students learns at their own pace (auto-regulated learning). When used in a classroom, it allows students to independently validate their understanding, thereby decreasing the need for assistance from the instructor. When used in distance learning, students enjoy the flexibility of choosing the time, place, and conditions most suited for studying. Indeed, our experiments showed that students worked (i.e. accessed the tool) from various places and at greatly varying times, including evenings and nights.

Similar to certain program assessment environments or plugins, the tool adopts a hands-on approach. Yet, it allows students to submit incomplete answers and still obtain feedback, thereby gradually building upon, and improving, their answers. For each proposal, students obtain an overall *score* (a success rate, rather than a *grade*) and are granted additional attempts to improve it. This imparts a game-like quality to their training, as reported by a significant number of students. Even Advanced students are indirectly driven to this friendly competition of finish exercises before others do.

Although we believe ShellOnYou has many advantages, it also suffers from two main limitations. Firstly, students need to be minimally familiar with tarball archives: each exercise comes with an accompanying tree structure directory of files to manipulate, and each solution has to be submitted in the form of an archive. We remedy this by proposing a 10-minute lecture session and a 30-minute tutored lab to less advanced students. The second limitation results from our ambition to provide students with a actionable feedback. This requires a careful analysis of the archive they submit in their an answer. In particular, when expecting answers to be written in a given text file, care must be taken when analyzing its lines, as students can incorporate the correct answer in the midst of a sentence or a paragraph, or with unexpected spelling. Nevertheless, the number of cases to consider can be drastically reduced by giving them a formatted answer file to be completed, along with precise instructions. This, in turn, allows to greatly reduce the size of the Python script required to analyze student answers. Yet, such scripts often comprise between 200 and 300 lines due to the need to examine several elements: indeed, active exercises usually contain three to ten questions, each one leading to one or several *tests*. Though these scripts contain boilerplate code, generating and proposing a new exercise (including making it cheat-proof) can take up to two working days. As long and costly as this initial investment may seem, thusly automating the process streamlines the chores of performing manual inspection, producing written feedback and composing an email for each solution submitted by students for every year the exercise is accessed. In our case, this proved a cost-effective approach from the very first year. Moreover, a Python library was made available to handle repetitive parts, and instructors could access the scripts of existing exercises to re-use code and learn from examples.

3 METHODS

3.1 The learning situation

Over the past four years, we set up the same learning situation for cohorts of 40 to 50 students at the beginning of our OS course. In the first week of their curriculum, students typically take 4.5 hours of introductory lectures on Unix systems – including approximately 1 hour on shell commands, combined with 4.5 hours of lab sessions during which students with little prior Unix knowledge are tutored in a classroom, while other students work autonomously. Links to external resources (textbooks, websites, command dictionaries, etc) are also provided to students during the lectures and collected on an LMS course page, so that students can refer to them when looking for complementary information or help on the CLI. On the last day of that initial week, students are instructed to register and log into the ShellOnYou tool, and are given a basic exercise to familiarize themselves with the tool's environment. In the following week, ShellOnYou generates a new daily exercise: students have 24 hours to submit their answers - as many as they wish within that period. Each student submits answers separately ; indeed, student statements are sometimes individualized, and one of our educational goals is to ensure that each one of them improves upon, or re-activates, targeted practical knowledge items. However, we also explicitly invite them to work in teams: we induce them orally to do so, book lab rooms so they gather physically and collaborate outside of class hours, and we also make sure their schedule allows for shared free time during the study week. Furthermore, we also encourage students to continue their collaboration via social networks. Indeed, the 24-hour period they are given to submit answers allows them to work at the time of their choice, or to adapt to the availability of other students. The OS course then continues for the rest of the term and covers other topics than the CLI, but students are able to easily build upon the foundational practical knowledge they have acquired over this initial two-week period.

3.2 Assumptions at the end of the learning situation

We tried to measure whether the ShellOnYou tool and the learning situation allowed us to meet our primary objectives (see end of Section 1). Our students had already completed a 2-year undergraduate program. We made four assumptions on our student after their initial 2-week training period, as detailed in the previous section:

(1) Students expanded their procedural knowledge of the Unix CLI (hard skills);
(2) Solution-driven group dynamics emerged;
(3) Students perceive their procedural knowledge has expanded
(4) Students can identify factors that facilitate their learning.

Table 1: Profiles of students in the cohorts studied.

Cat.	2018	2019	2020	2021
A	16	10	15	15
B	21	25	23	23
C	1	3	4	1
D	8	4	4	2
Tot.	46	42	46	41

The categories include students, who, depending on their prior curriculum, (A) were familiar with Unix systems, (B) had had medium to limited exposure to Unix, (C) had had limited exposure to Unix, and (D) had no prior Unix knowledge.

3.3 Feedback collection and analysis methodology

We based our work on the Design-Based Research (DBR) paradigm [24, 27]. This approach makes it possible to instantiate theoretical models in the form of digital/computer applications in which identified users carry out contextualized uses. The results that we present here reflect the training system as assessed for four cohorts over successive academic years.

Table 1 shows the background heterogeneity of students entering the program. Students in the B category followed a supposedly similar undergraduate curriculum with courses in various scientific fields, but with a large variance in practice: depending on their university, they took one to four CS courses (mainly computer literacy and programming courses). Finally, some students (D category) had never studied Unix before.

We followed a classic didactic engineering methodological approach based on the Theory of Didactic Situations (TSD) [8]. During the evaluation, we asked the students to fill a survey, which they accessed via their LMS course page. We clearly stated that this was an anonymous survey, and that the LMS activity was set up accordingly. The goal of this survey was to collect our students' impressions and comments on their experience in the learning situation organized for their initial 2-week training. The survey comprises 11 closed questions and 2 open questions. The answer rates were 52%, 48%, 57% resp. 59% for the 2018 to 2021 years. Lastly, we performed an analysis of the ShellOnYou database, which yielded learning analytics for the last two cohorts. Before using the tool and answering surveys and interviews, the students were informed that the data collected would be used for research purposes; they gave their consent for this use.

We carried out a quantitative analysis (closed questions), as well as a qualitative analysis (open questions). Combining the two allowed us to highlight the learning situation components as well as the strategies implemented (learner behavior). We analyzed the open questions in two phases. First, we analyzed the statements made in the students' answers to the survey thanks to QDA Miner Lite software (v2.09) [15]. Its text segment encoding system allows each corpus element to be labeled independently and according to categories that emerge during the encoding. Categories first appeared individually; they were then grouped; finally; subcategories were identified and defined.

4 RESULTS

In addition to data from Table 1, a closed question in our survey was designed to collect information on how students evaluated their Unix knowledge at the time of answering: ≈ 11% had never heard of Unix before; ≈ 14% were very nervous about studying this system; ≈ 34% were a bit apprehensive about studying this system; ≈ 33% were familiar with this system, and ≈ 12% were quite confident with using it. Though only half of the students submitted their answers to this question (recall return rates), this data nevertheless confirms the profile heterogeneity of our students.

4.1 Students expanded their procedural knowledge

As previously mentioned, ShellOnYou exercises are designed so they can readily be solved by students: each exercise provides both initial and feedback clues to its solution. Additionally, instructors' encouragements to share the learning experience also fosters student performance. As a result, almost all the students in our four cohorts regularly achieved maximum scores. Specifically, the learning analytics indicate that, on average, 99.4% (2020), resp. 99.8% (2021), of all the students obtained the maximum score for all the exercises they solved. Interestingly, these high final success rates contrast with the fact that, on average, only 14.3% (2020), resp. 20.54% (2021) of the students obtained a maximum score on their first trial for each exercise. This seems to indicate that students improved their procedural knowledge in the process of solving the exercises.

Another indirect hint that they improved their skills is that 64% (in 2018), 70% (2019), 42% (2020), 72% (2021) of them reported that they had benefited from the help of other students to solve exercises. These score improvements are therefore partly explained with student interactions, which confirms that peer-learning took place, helping both less capable students increase their knowledge, and more knowledgeable ones consolidate theirs.

Furthermore, the learning analytics show that, on average, 20.93% (in 2020) to 25.58% (in 2021) of the students went on proposing solutions to exercises for which they had already obtained the maximum score. Besides evidencing genuine motivation, this seems to indicate that these students spent extra time exploring other ways to perform the tasks required by exercises, maybe in part as a result of peer interactions. The Unix playground has the potential for creating this type of scenario, since shell commands and mechanisms such as redirections and pipes often provide different paths to reaching a similar goal. This kind of practice can only foster the acquisition of procedural knowledge of such systems.

Lastly, in 2021, when presented with a set of 10 procedural knowledge, on average, 38.8% of the answering students stated that they had prior knowledge, and 78.2% that they had acquired it as a result of the learning situation.

4.2 Emerging Group Dynamics

In our experiments, all students - including those already familiar with Unix - were asked to solve several exercises. One of our

Table 2: Types of assistance interactions between students.

Cat.	2018	2019	2020	2021
Does not apply	39%	30%	62%	24%
Question explanation	11%	15%	4%	8%
Error explanation	18%	15%	12%	48%
Answer explanation	29%	30%	23%	4%
What to type as an answer	4%	10%	0%	12%

Answers to the survey question:*When another student helped you, what sort of interaction assistance did you usually get?* The "*Does not apply*" option stands for students indicating they usually solved exercises without help.

goals was to generate student interactions, and, more specifically, support dynamics between our *resource* students and the less capable students in our cohorts. We therefore designed the exercise statements and the feedback so they would provide unusual flags or additional commands for students to further explore the questions or even arrive at answers in creative and convoluted (i.e. mysterious) ways. This built-in trick effectively helped us keep the *resource* students in the lab rooms with the less capable students, who struggled with their own attempts at solving the exercises. This indirect commitment obligation promoted mutual support interactions: experienced students reported staying in the room after having solved their own exercises to help other students with theirs. More than 50% of the students received help from other students. Table 2 shows the various types of assistance received. The data indicates that student support interactions most often involved explanations. This constitutes traces of group dynamics emerging from the learning situation where very few students initially knew each other. In video interviews (data not shown), 2021 students reported that most students started working on their own, but sought help through various communication channels when faced with unresolved challenges. These group dynamics are critical : indeed, Booth states that "*The experience of learning in a group and the stage of maturity as a knower appear to be closely related*" [7].

4.3 Students' self-evaluation of their progress

The results reported here were obtained by analyzing the students' answers to the two open questions of the anonymous survey: "*What are the positive aspects*" (resp. "*negative aspects*") "*of this 2-week learning situation experience?*". Overall, 224 *student items* were identified and categorized with *QDA Miner Lite* software. A *student item* is defined by any word or group of words students used to answer the above questions. Table 3 shows the distribution of student items across three categories. The last two rows directly relate to assumptions (3) and (4).

Student items included in the *General aspects* of Table 3 indicate, for instance, that students liked the learning situation, finding it *e.g.*, "*globally positive*" (5/25 items), "*fully playing its role*" (3/25) and that "*positive aspects outweigh negative ones*" (2/25). Negative comments indicate, *e.g.*, that the exercise week required a consistent study effort, unlike the following labs of the OS course (2/25).

Table 3: Learning situation aspects raised by students items.

Type	# Student items
General aspects	25 (11.16%)
Knowledge acquisition	79 (35.26%)
Facilitating factors	120 (53.57%)

Answers to the survey question:*What are the positive and negative aspects of this 2-week learning situation experience?*

Further analyzing the items falling in the *Knowledge acquisition* category of Table 3, we found that the perception the students formed of the learning situation could be detailed more precisely (Table 4). For instance, students identified the following strategic *individual* benefits: "*review the basics*" (7/30 items), "*remember commands*" (6/30), "*get back on track*" (5/30), "*learn and get familiar with Unix commands*" (7/30), "*train*" (1/30), or "*assess oneself*" (1/30). The first three of these items stem from the knowledge reinforcement concept; thus, they mostly originated from students with prior knowledge of the Unix CLI. Though this represents only 18 out of the 224 items reported by the students, they identify knowledge reinforcement over other individual strategic benefits in their reflection on the learning process. Indeed, reflective thinking constitutes a critical learning factor: "*reflection is linked to elements that are fundamental to meaningful learning and cognitive development*" [22].

As for strategic *collective* benefits (5 items), students listed "*catching up*", or "*leveling*" (4/5) and "*teaching the basics to all students*" (1/5). Clearly, several students were aware of their cohort's heterogeneity; some even perceived the usefulness of building a learning community. Unfortunately, our survey questions did not collect the data which would have allowed us to further analyze this aspect.

The *Confidence* entries in Table 4 show that the learning situation reassured students on four levels: self-confidence ("*I could gain self-confidence* ", "*I feel more independent/autonomous*");"*skills*" or procedural knowledge acquired; acquired conceptual knowledge ("*Now I know basic Unix commands*" / "*new commands*", "*I know how to use Unix now*"); usefulness of this practical knowledge.

Students also identified the *Assessment method* as a knowledge acquisition factor, pointing to the playful aspects of the tool (4/10 items), *i.e.*, getting hints in the feedback, being able to improve their scores ("*it is challenging*", "*I learned without having the impression to work at it, which is the most important*"). The fact that we used *scores* (from 0 to 100%) as in games rather than grades also seemed to motivate them (3/10); they also liked being given several attempts at solving exercises to improve their performance. This highly correlates with the learning analytics showing than 85% (2020), resp. 79% (2021), of them improved upon their first-attempt scores.

Some students stressed the efficiency of the learning pace (3/9 items) ("*allows to quickly learn basic commands*" with a "*a rather efficient learning pace for me as beginner*"). They highlighted the scheduling freedom afforded by the tool (3/9) ("*we can work when we want*"). The duration (one week) of the exercise part seemed adequate to some students (3/9) ("*Doing exercises over one week to hone our skills is a very good thing*").

Table 4: Knowledge acquisition perceived by students.

Category	# Student items	Inner distrib.
Strategic benefit	**35 (44.30%)**	
individual		85.71%
collective		14.28%
Confidence	**25 (31.65%)**	
self-confidence		32%
owning skills ("*I can*")		32%
owning knowledge ("*I know*")		28%
learning usefulness		8%
Assessment method	**10 (12.66%)**	
playful aspects		70%
scores		30%
Learning pace	**9 (11.39%)**	
efficiency		33.33%
time freedom		33.33%
1-week exercise period		33.33%

Knowledge acquisition perception from student answers to the survey. For each category, the number (and percentage over 79) of student items is indicated, together with the relative percentage of each related subcategory.

Table 5: Facilitating factors identified by students.

Category	# Student items	Inner dist.
Exercise components	**79 (65.83%)**	
difficulty level		22.78%
exercise type		22.78%
exercise timing		20.25%
assessment modality		18.99%
statement formulation		15.19%
Assistance	**19 (15.84%)**	
support interactions		57.89%
hints contribution		42.11%
Technical aspects	**14 (11.66%)**	
tool usefulness		28.59%
access device		21.42%
availability		21.42%
remote access		21.42%
special characters		7.15%
Availability of lab rooms	**8 (6.66%)**	

Categories of facilitating factors reported by students. For each category, the number (and percentage over 120) of student items is indicated, together with the relative percentage of each related subcategory.

4.4 Facilitating factors identified by students

We identified 14 factors in the student items relating to assumption (4): the identification by students of factors influencing their learning. We grouped these factors into four categories (Table 5). A significantly recurring factor (18/79) is the difficulty level of the exercises: the students reported that they were too easily solved. On the other hand, the students praised their progression (11/79), highlighting once more the heterogeneity of our cohorts; but it also hints that our exercises were more aimed at novices than knowledgeable students. Indeed, at this point in the OS course, our main goal was to ensure that *all* students would acquire *minimal* procedural knowledge. Concerning the assessment modalities, students praised being given several attempts to solve exercises and receiving feedback on the solutions they submitted (9/79). They appreciated the assistance received from both student support interactions (11/19) and clues provided in the exercise statements and feedback (8/19). Finally, with regard to technical aspects, students hesitated when selecting devices to connect to the tool (3/14), or, in contrast, found it easy to access (4/14).

5 CONCLUSION

We introduced a new web-based exercise tool to promote the acquisition of practical knowledge of the Unix system; we presented an in-depth analysis of the tool's use in a learning situation with four student cohorts. The tool offers a short list of auto-grading exercises, each centered on specific procedural skills. After submitting their work, students get detailed feedback on their answers, including explanations of some of their errors, clues to improve their performance, and a score. Instructors can set up sessions for their students on our tool's deployed instance, and propose new exercises there. However, this instance has limited computing resources and should be used mainly for testing the tool's functionalities; institutions are encouraged to deploy their own instance – the tool is distributed freely. Moreover, the tool can be integrated with a Learning Management System to create seamless activities for students.

The analysis of our learning situation shows that the tool actually allows students to improve their procedural knowledge. Our results also showed that the two-week learning situation and the tool combined to promote the emergence of educationally-rich group dynamics. Moreover, the learning situation and the survey that we invited students to fill helped them develop metacognitive capabilities [10, 26], *i.e.*, their ability to reflect upon their thinking process and to self-evaluate [22].

As a whole, these analyses lead us to consider this learning situation as an *enabling environment*, *i.e.*, a technical and social environment providing students with the opportunity to develop new procedural knowledge and skills, to increase their action possibilities and degree of control over their tasks, and to widen their operating methods, i.e. their autonomy [12].

ACKNOWLEDGMENTS

This work was funded by *Fondation Polytech*, the *Polytech Network* through the *IDEFI AVOSTTI* ANR project and the *Opening* project (call *Hybridation des formations d'enseignement supérieur* of the French Ministry of Higher Education, Research and Innovation). It was also supported by the CNUMF of *Université Montpellier* and by the Polytech IT Department. In particular, we thank Luca Cimini.

REFERENCES

[1] [n. d.]. https://www.codingame.com/. Online; accessed September 2018.

[2] [n. d.]. https://tech.io/. Online; accessed September 2019.

[3] [n. d.]. https://www.hackerrank.com/. Online; accessed November 2021.

[4] [n. d.]. https://www.pluralsight.com/codeschool. Online; accessed November 2021.

[5] [n. d.]. https://www.katacoda.com/. Online; accessed January 2021.

[6] John Biggs. 1996. Enhancing Teaching Through Constructive Alignment. Higher Education 32 (10 1996), 347–364. https://doi.org/10.1007/BF00138871

[7] Shirley Booth. 2001. Learning Computer Science and Engineering in Context. Computer Science Education 11, 3 (Sept. 2001), 169–188. https://doi.org/10.1076/csed.11.3.169.3832

[8] Guy Brousseau. 1988. Théorie des situations didactiques. La Pensée Sauvage, Grenoble.

[9] Stephanie M. Doane, James W. Pellegrino, and Roberta L. Klatzky. 1990. Expertise in a Computer Operating System: Conceptualization and Performance. Human–Computer Interaction 5, 2-3 (June 1990), 267–304. https://doi.org/10.1080/07370024.1990.9667156

[10] Anastasia Efklides. 2001. Metacognitive Experiences in Problem Solving. Springer Netherlands, Dordrecht, 297–323. https://doi.org/10.1007/0-306-47676-2_16

[11] Christian Estler and Martin Nordio. [n. d.]. https://codeboard.io/. Online; accessed January 2018.

[12] Pierre Falzon. 2005. Ergonomics, knowledge development and the design of enabling environments. In HWWE 2005 : humanizing work and work environment : proceedings of the International Ergonomics Conference. New Delhi : Allied Publishers, Guwahati, India, 1–8.

[13] Ricardo Hoar. 2014. Generally Educated In The 21st Century: The Importance Of Computer Literacy In An Undergraduate Curriculum. In Proceedings of the Western Canadian Conference on Computing Education (WCCCE '14). Association for Computing Machinery, New York, NY, USA, Article 6, 5 pages. https://doi.org/10.1145/2597959.2597964

[14] Tyson Kendon and Ben Stephenson. 2016. Unix Literacy for First-Year Computer Science Students. In Proceedings of the 21st Western Canadian Conference on Computing Education (WCCCE '16). Association for Computing Machinery, New York, NY, USA, Article 14, 4 pages. https://doi.org/10.1145/2910925.2910930

[15] R. Barry Lewis and Steven M. Maas. 2007. QDA Miner 2.0: Mixed-Model Qualitative Data Analysis Software. Field Methods 19, 1 (Feb. 2007), 87–108. https://doi.org/10.1177/1525822x06296589

[16] Richard Lobb and Jenny Harlow. 2016. Coderunner. ACM Inroads 7 (02 2016), 47–51. https://doi.org/10.1145/2810041

[17] Serghei Mangul, Lana S. Martin, Alexander Hoffmann, Matteo Pellegrini, and Eleazar Eskin. 2017. Addressing the Digital Divide in Contemporary Biology: Lessons from Teaching UNIX. Trends Biotechnol 35, 10 (10 2017), 901–903.

[18] Robert McCormick. 1997. Conceptual and Procedural Knowledge. International Journal of Technology and Design Education 7, 1-2 (Jan. 1997), 141–159. https://doi.org/10.1023/a:1008819912213

[19] Gaëlle Molinari, Bruno Poellhuber, Jean Heutte, Elise Lavoué, Denise Sutter Widmer, and Pierre-André Caron. 2016. L'engagement et la persistance dans les dispositifs de formation en ligne : regards croisés. Distances et médiations des savoirs (online) 13 (2016), Online. https://doi.org/10.4000/dms.1332

[20] Matthieu Moy. 2011. Efficient and Playful Tools to Teach Unix to New Students. In Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education (ITiCSE '11). Association for Computing Machinery, New York, NY, USA, 93–97. https://doi.org/10.1145/1999747.1999776

[21] Bruno Poellhuber, Normand Roy, and Ibtihel Bouchoucha. 2019. Understanding Participant's Behaviour in Massively Open Online Courses. The International Review of Research in Open and Distributed Learning 20, 1 (Feb. 2019), 221–242. https://doi.org/10.19173/irrodl.v20i1.3709

[22] Carol Rolheiser, Barbara Bower, and Laurie Stevahn. 2000. The portfolio organizer: Succeeding with portfolios in your classroom. ASCD, Alexandria, Virginia USA.

[23] Marek Suppa, Ondrej Jariabka, Adrián Matejov, and Marek Nagy. 2021. TermAdventure: Interactively Teaching UNIX Command Line, Text Adventure Style. In ITiCSE 2021: 26th ACM Conference on Innovation and Technology in Computer Science Education, Virtual Event, Germany, June 26 - July 1, 2021, C. Schulte, B.A. Becker, M. Divitini, and E. Barendsen (Eds.). ACM, Virtual event, Germany, 108–114. https://doi.org/10.1145/3430665.3456387

[24] The Design-Based Research Collective. 2003. Design-Based Research: An Emerging Paradigm for Educational Inquiry. Educational Researcher 32, 1 (Jan. 2003), 5–8. https://doi.org/10.3102/0013189x032001005

[25] Kristina von Hausswolff. 2017. Hands-on in Computer Programming Education. In Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17). Association for Computing Machinery, New York, NY, USA, 279–280. https://doi.org/10.1145/3105726.3105735

[26] Lev S. Vygotsky. 1978. Mind and society: The Development of Higher Mental Processes. Harvard University Press, Cambridge, MA. http://www.learning-theories.com/vygotskys-social-learning-theory.html

[27] Feng Wang and Michael J. Hannafin. 2005. Design-based research and technology-enhanced learning environments. Educational Technology Research and Development 53, 4 (Dec. 2005), 5–23. https://doi.org/10.1007/bf02504682