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

In this paper we present both a new tool for computing education and an analysis of its use over four successive student cohorts.

The tool was developed to help instructors manage large numbers of students learning the Unix system. It is an autonomous web app that can also be integrated in an LMS platform with the LTI protocol. It offers exercises combining specific practical knowledge about using a Unix-like operating system from the command line. The basic principle is that students can submit as many answers they want to an available exercise and get each time both a score and a specific feedback, almost instantly. In practice, this encourages students to resubmit and ultimately improve their procedural knowledge. The learning tool can also deliver slightly individualized statements, hence favoring situations where students grow in skills by combining personal research and peer learning. Being online, this tool offers flexibility to students and it naturally fits distance learning programs or periods. We found it particularly useful in a context of students with heterogeneous prior knowledge.

Successive cohorts were proposed a learning situation involving the tool. We combine qualitative and quantitative methods to analyze their answers to surveys. We aim at characterizing their rise in procedural knowledge and the building of a group dynamics. Several emerging dimensions are: benefits of the tool and the learning situation, reassurances in their engagement, playful aspects and grading system as a source of motivation in their daily work, the chosen learning pace. These are characteristics of a learning by doing practice, but we also noticed the importance to encourage their reflexive and meta-cognitive processes.

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, automated feedback, student 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 (ITiCSE '2022)*. ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

ITiCSE '2022, July 08–13, 2022, Dublin, Ireland

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

A Unix introductory course aims 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]. Such a course appears in almost each Computer Science program, usually in its first academic year. Students from many other fields (mathematics, physics, natural and social sciences, *e.g.*, see [16]) are also concerned by such courses due to the predominance of the Unix system in computing facilities. As a result, many universities have to tutor large cohorts of students that 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 to computer science (CS) careers by proposing a program they follow from their third to fifth academic year to obtain an engineering degree. We propose an Operating System (OS) course right at the beginning of their first term in our curriculum. This course starts by presenting students with material on a Unix system, the CLI being at the forefront. Indeed, as in other CS curriculum, knowing one's way on this system is a prerequisite to many other courses and students with a deficit on this topic will have problems in graduating in the end. Students entering our curriculum have an heterogeneous background, some having never eared of Unix systems, most have used it in a strictly minimal way during a programming course in a multidisciplinary curriculum, while some have used it on a regular basis in a purely CS undergraduate curriculum. This echoes the observations of Moy [19] in a similar context. Moreover, very few students know each other at first since they reach us by succeeding

at a national examination procedure irrigating our network of engineering schools. Furthermore, for various institutional reasons, we have a relatively small number of instruction hours to dedicate to this topic. On the other hand, we only handle cohorts of 40 to 50 students, while it is common that an introductory OS course is also proposed in programs followed by several hundred students. In the latter case, another difficulty is to find enough qualified instructors. This is pushed to an extreme in computer literacy courses [12] where survival skills on a Unix system are sometimes sought for, and that concern all first year students entering a faculty.

To favor the learning of practical knowledge of the Unix CLI in this context, we needed a dedicated learning situation. Among other learning approaches, the hands-on approach most often obtains good feedback from students and is advocated for being efficient and well suited to engage students [18, 21, 25]. Thus, we wanted practical activities to be at the center of the learning situation we would design. However, this approach tends to have the annoying particularity to generate a large number of student works to be examined by instructors. Among others, Kendon and Stephenson [13] provide hands-on instruction on the use of the Linux CLI and report that *the single biggest challenge that the instructors faced with this course was the volume of work that needed to be graded.* Having the possibility to only spend a few hours with the students, we aimed at finding a tool to examine students' proposals in an automated way. Though the Unix CLI can be envisioned in a programming perspective (shell scripts), we believe it can only be grasped by using it interactively in a Unix environment. Following Doane et al. [8], we want to *measure the students' performance in tasks requiring them to comprehend and produce Unix commands.* For this reason, assessment websites or plugins in learning management systems (LMS) plugins centered on programming skills were ruled out of the tools we could rely on [10, 15]. Only a small number of hours being funded, we could not engage ourselves neither in paid plans of websites which also ruled out e-recruitment platforms [1–3] and publishers' tools [4, 20].

Not finding any tool on which we could rely on we decided to develop a web-based tool tailored to our needs and in particular allowing us to provide automated feedback to students learning the Unix CLI. This tool, called ShellOnYou, will be freely distributed. It can be easily installed as an autonomous website thanks to the docker technology. Moreover, its exercises can also be integrated as activities in an LMS, leading to a seamless use. We describe it in more detail in Section 2 below.

Having such a tool at our disposal, we then integrated it in a learning situation with two underlying goals: *i)* mainly, obtaining a students' rise in practical, *i.e.*, *procedural* [17] knowledge (hard skills) on the Unix CLI but also *ii)* the constitution of a group dynamics that will benefit the cohort in the subsequent learning phases of the curriculum (in our case, students have 80% of their courses in common during the 3 years to graduation). The learning situation is described below in Section 3.1.

We applied this situation on four successive student cohorts, in the fall term, from 2018 to 2021. We questioned the learning situation by analyzing students' answers to surveys and interviews. Our main focus question is: "*How such a tool and learning situation favors the learning of practical knowledge in computer science?*", keeping in mind that we handle students with heterogeneous backgrounds and have few hours and instructors at our disposal. Section 4 reports our findings in this direction when analyzing students' answers.

2 THE TOOL AT THE CENTER OF THE LEARNING SITUATION

2.1 Principles of the exercising platform

Instructors propose *exercises* which consist of a template (*i.e.*, generic) statement (plain text or HTML format), a template archive (*i.e.*, a tarball), and two Python programs: a code to randomly individualize student statements and a code to analyze their answers. This template approach allows instructors to give slightly different versions of the same exercise to different students. By providing them with different versions, instructors handling student groups limit plagiarism as a simple copy/paste approach will then be unfruitful. The goal is not at all to impede communication between students, but rather to ensure it leads to an interaction between peers on the understanding of the instructions, to share useful tips and tricks to solve the exercise, instead of exchanging final complete solutions. In our experiments, we actually encouraged students to share thoughts and work next to one another, even booking a lab room for them, or referring to social networks for exchanges. This both helped knowledge transfer between peers and the appearance of a group dynamics.

Students access exercises on the ShellOnYou web platform by registering to an exercising *session* settled by their instructor. Such a session is available for a defined time period and possibly requires a password to enter. A session consists of an ordered list of exercises that the instructor can make available all at the same time, or one exercise a day, or only when obtaining a score (success rate) above some threshold to the previous exercise.

To solve an exercise, a student first downloads a tar archive consisting of a small file structure to explore while following instructions given on the exercise page. These instructions ask him/her to perform some tasks in this file structure or to complete an answer file it contains. Then the student submits his/her work by creating an archive of the modified file structure and uploading it on the ShellOnYou platform. The platform then gives a detailed feedback to the student on the different points mentioned in the exercise statement, and provides him/her with an overall score. The feedback informs the student on his/her particular misconceptions and wrong answers. The student can then learn from previous errors and come back to his/her work and later submit an improved answer, and so on until the exercise is not available anymore (when reaching the deadline of the exercise or of the session). Obtaining a personalized feedback allows students to gradually improve and ultimately reach the planned skills. Of course, this happens when the exercise level is aligned with the tutoring experience delivered outside the platform [5]. From our experiments, the overall score they obtain on each attempt, combined to the possibility to retry within some minutes is quite engaging for students as they get caught in a game-like spiral, all the more they exchange on their respective scores. In our experience, this works best when the exercise is decomposed in several intermediary questions and when a student can obtain a first attempt within a few dozen minutes.

The feedback given to students is obtained by running the code proposed by the creator of the exercise, which behaves as a series

of unit tests in the software development industry (an approach common with the *Code runner* plugin [15]). Usually, instructors know common pitfalls they have to look for in students answers to a particular exercise they consider. Transferring this knowledge into the code that analyses students answers allows their expertise to be useful to hundreds of students at the same time or to time and places where instructors are not available, yet when and where students are ready to improve. And indeed, our experiments reveal that students work in various different places and at greatly varying time slots, including evenings and nights.

The tool provides instructors with basic analytics. For each session they organize, instructors can know which student submitted an answer, when this happened and can access the obtained feedback and score.

2.2 Advantages and drawbacks of the ShellOnYou tool

ShellOnYou favors a process where each student learns at his/her own pace (auto-regulated learning). When used in a classroom, this allows students to test their understanding of concepts by themselves and validate knowledge on their own, giving the instructor more time to assist students having difficulties. Moreover, students benefit with ShellOnYou from the flexibility offered by online tools. They can then choose the most suited time, place and conditions for them to work, as all that is required is an internet connection and most students nowadays have access to one *e.g.*, by sharing their smartphone connection.

ShellOnYou shares objectives with other tools allowing a hands-on approach to CS, such as program assessment environments or plugins. Yet, ShellOnYou allows students to submit incomplete answers and still obtain feedback, thus allowing them to gradually build and improve their answers. In comparison, programming assessment tools usually give a frustrating null grade to a student' proposal containing only a single syntax error, not accounting for correct parts in the proposal. In comparison, ShellOnYou offers instructors the possibility to detect any positive element.

The fact that a students obtain an overall *score* (a success rate, rather than a *grade*) and are offered multiple attempts to improve on their score gives their training a game-like flavor (as a significant number of them reported in the final survey). Even advanced students are driven forward by these elements, competing to finish their current exercise before other advanced students.

Although we think ShellOnYou has many advantages, it suffers from two main drawbacks. The first one is that students need at least a minimal familiarity with tarball archives (each exercises comes with an accompanying tree structure of files to manipulate while solving the exercise, and that each solution has to be submitted in the form of an archive). Though GUI of Unix systems nowadays allow to make one's way out of this problem by only a few mouse clicks, this prerequisite needs some attention. We chose to propose 10mn lecture time on this topic to our less advanced students and proposed them a 30mn lab with the presence of an instructor.

A second drawback of the current system is a direct consequence of the will to propose students with a useful feedback of their answers to exercises. This requires a careful analysis of the archive they submit as an answer. In particular, when expecting some answers to be written in a given text file, care must be taken when analyzing its lines as students can incorporate the expected answer in the midst of a sentence or sometimes a paragraph, or with a spelling different from that expected, or enclosed in a variable number of surrounding spaces or with astonishing punctuation characters. Though, the worst nightmare can be avoided and the number of situations be drastically reduced by proposing in the question archive an answer file to be completed by the student, jointly with precise instructions in the exercise statement. This special care allows to greatly reduce the size of the Python script analyzing answer archives submitted by students. Yet, such a script often still contains between 200 and 300 lines due to the need to examine different points: current exercises usually contain three to ten questions, each one leading to one or several *tests*. For each test, a code aiming at a useful feedback needs to consider various common student mistakes as well as distinguish correct answers from partially correct ones. Though such a script contains some boilerplate code, the proposal of a new exercise can take up to two working days including making it bullet-proof. This investment must be put into perspective as it avoids manual inspection, written feedback and email by an instructor to each attempt submitted by each student over each year the exercise is used. In our case, this was cost-effective from the first year as each of the 46 students we tutored did 2.28 attempts per exercise on average. Moreover, to lessen the burden of proposing a new exercise, a Python library is made available to instructors that handles the repetitive parts. Finally, instructors can access scripts of existing exercises to re-use some code and learn from examples.

3 METHODS

3.1 The learning situation

For the past four years we settled the same learning situation at the beginning of the OS course, proposed to cohorts of 40 to 50 students. During the first week of their curriculum, students attend 4.5h hours of introductory lectures on Unix systems – among which roughly 1 hour is dedicated to shell commands, interleaved with 4.5h hours of lab sessions during which students with low to poor prior Unix knowledge are tutored in a classroom, while other students work in autonomy. Links to external resources (textbooks, websites, command dictionaries, etc) are also provided to students during the lectures and reported on an LMS course page, so that they can refer to them when looking for complementary information or help on the CLI. The last day of the week they are asked to register on the ShellOnYou tool and provided with a basic exercise whose only role is to allow them to get familiar with the way the tool works. The following week, ShellOnYou makes a new exercise available to them every working day and they are given 24 hours to do it, being allowed to submit as many answers they want during this time lapse. Each student submits answers separately (and indeed student statements are sometimes individualized), as the goal is that each one of them improves or re-activates the targeted practical knowledge. Though, we explicitly encourage them to work in group, by oral inducement, by booking a lab room they can use outside of class hours, and by making sure that every day they have a free common time slot during working hours. Though,

Table 1: Profiles of students from studied cohorts.

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

Categories indicate students that depending on their prior curriculum (A) are familiar with Unix systems, (B) had moderate to no exposure to Unix, (C) had low to no exposure to Unix and (D) never met Unix previously.

students also exchange by social networks. Indeed, the 24h delay they have to produce answers allows them to work at the time they prefer or to adapt to the availability of other students. The OS course then continued for the rest of the term on other topics than the CLI but building on the practical knowledge they have at the end of this two weeks period.

3.2 Assumptions at the end of the learning situation

We tried to measure whether the ShellOnYou tool and the chosen learning situation answer our primary objectives (as exposed at the end Section 1) or sometimes exceeds them. We postulate four assumptions on student cohorts after they lived the two first weeks of their program, orchestrated as detailed in the previous section:

(1) Students have increased their level of procedural knowledge on the Unix CLI (hard skills);
(2) a group dynamics has been created to solve problems;
(3) Students perceive the improvement of their procedural knowledge;
(4) Further, students can identify facilitating factors that inlfuence their learning.

3.3 Collected feedback and analysis guideline

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 that can be used by identified actors who will carry out contextualized uses. The results that we present reflect the training system as it has been evaluated over four cohorts of successive academic years.

Table 1 indicates the profiles of students we hosted during the four years of the study. Students of the B category are difficult to sort out: they followed a supposedly similar undergraduate curriculum with courses from various scientific fields but with a large variance in practice: depending on their host university during these study years, they took from one to four CS courses (mainly computer literacy and programming courses). Besides, only some of them have been confronted to the Unix system before.

We followed a classical methodological approach of didactic engineering based on the Theory of Didactic Situations (TSD) [7]. During the evaluation phase, we proposed students with as survey available on the LMS course page. It was clearly stated that this was an anonymous survey and the LMS activity was settled accordingly. The goal of this survey was to collect students' feelings and comments on the learning situation they lived for the two first weeks of their new curriculum. The survey contains 11 closed and 2 open questions. The return rates are 52%, 48%, 57% resp. 59% for the 2018 to 2021 years. Lastly, an analysis of ShellOnYou's database provided us with learning analytics for the last two cohorts. Before using the tool as well as answering surveys and interviews, students were informed that collected data would be used for research purposes, and they approved this use.

We carried out a quantitative analysis of these data (closed questions) associated with a qualitative analysis (open questions). The combination of these two analyzes makes it possible to highlight the learning situation components as well as the implemented strategies (learners' actions). The analysis of the open questions took place in two phases. First, we analyzed the statements made by students answers to the survey thanks to the QDA Miner Lite software (v2.09) [14]. Its text segment encoding system allows each corpus element to be labeled, independently, according to categories that emerge during the encoding. Categories first appeared in isolation, then were grouped, leading to the definition of subcategories.

4 RESULTS

A closed question of the survey informs us on how answering students situate themselves, which complements data from Table 1: $\approx$ 11% had never heard of Unix before; $\approx$ 14% were greatly worrying to study this system; $\approx$ 34% felt a bit of apprehension to be confronted to it; $\approx$ 33% were comfortable with this system, and $\approx$ 12% were quite confident with it. Though these perceptions concern only half of the students (recall return rates), they nevertheless confirm the heterogeneity of their profiles.

4.1 Students improve their procedural knowledge

Recall that exercises are designed to be succeeded by students, as each one contains hints on its resolution (both initially and in feedback to their first trials) and as we encouraged students to share their knowledge. Indeed, almost all students of the four cohorts regularly obtain a maximum score at each exercise they consider. In particular, the learning analytics indicate that 99.4% (2020), resp. 99.8% (2021), of all the students obtain a maximum score (on average over all exercises they considered). Interestingly, there is a contrast between these high final success rates and the fact that only 14.3% (2020), resp. 20.54% (2021), students obtained a maximum score on their first trial at an exercise on average. This seems to indicate that students improved their procedural knowledge in the process of resolving the exercises.

Another result that we see as an indirect hint that they improved their skills is that 64% (in 2018), 70% (2019), 42% (2020), 72% (2021) of them declare to have benefited form the help of other students for resolving some exercises. So the increased scores obtained at exercises are partly explained by exchanges between students, which indicates that some peer-learning took place, both helping less capable students to rise in knowledge and consolidating the learning of more knowledgeable ones.

Table 2: Types of assistance 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 me, what sort of interaction did you usually got?* The "*Does not apply*" option stands for students indicating they usually solved exercises without help.

Further, learning analytics show that on average 20.93% (in 2020) to 25.58% (in 2021) students went on proposing answers to exercises for which they already had reached a maximum score. Besides showing a clear motivation, this seems to indicate that these students (maybe predominantly at ease with the Unix CLI) spent time discovering other ways to achieve tasks asked by exercises, maybe in part as a result of exchanges between peers. The Unix playground has the potential for such a scenario as shell commands and mechanisms such as redirections and pipes often offer different paths to reach a same goal. This kind of practice can only improve their procedural knowledge of such systems.

Lastly, when presented (after the experience) with a set of 10 procedural knowledge, in 2021, 38,8% of the answering students thought they had such knowledge before the experience while 78,2% of them thought they such knowledge after. This twofold increase factor can be subjective but was also observed for the other cohorts.

4.2 A group dynamics is observed

During our experiments, all students were asked to solve the exercises, including those already familiar with Unix. As we wanted these *resource* students to stay in the lab room, exercises statements as well as feedback sometimes mentioned an unusual flag or complementary command to explore or challenged them to answer a question in a more tricky (or mysterious) way. Overall, this helped keeping them in the lab room while less capable students were struggling with their own versions of the exercise. This promoted mutual help and indeed experienced students reported to have often answered questions of other students, sometimes staying in the room after having solved their exercise.

We reported above that more than 50% students benefited form the help of other students. Table 2 reports on the type of assistance they received. This shows that students interactions about exercises most often involved explanations and not just what to type to answer correctly. These explanation phases are traces of a group dynamics building from the situation were very few students knew one another at the start of the term.

Six 2021 students relate in video interviews (data not shown) that most students started by working by themselves and sought help when blocked, through various communications channels.

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 the learning situation you experienced during these weeks?*

This group dynamics is important for us, since we follow Booth who states that "*The experience of learning in a group and the stage of maturity as a knower appear to be closely related*" [6].

4.3 Students perceive their progress

Results reported here are obtained by an analysis of the students' answers to the two open questions of the anonymous survey: "*What are the positive aspects*" (resp. "*negative aspects*") "*of the learning situation you experienced during these weeks?*". Overall, 224 *student items* were identified and categorized with the *QDA Miner Lite* software. A *student item* relates to any word or group of words a student has written to answer the above questions. Table 3 shows the distribution of student items between three categories we identified. The last two rows directly relate to assumptions (3) and (4).

Student items entering 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*" and that (3/25) "*positive aspects outweigh negative ones*" (2/25). Negative comments indicate for instance that the exercise week asked them a consistent work effort, unlike following labs of the OS course (2/25).

When analyzing further the student items falling in the *Knowledge acquisition* category of Table 3, the perception they have of the learning situation can be encompassed more precisely (Table 4). For instance, strategic *individual* benefits they identify are *e.g.*, "*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 three first of these items stem from the knowledge reinforcement idea, hence mostly originate from students with prior knowledge of the Unix CLI. Though this represents only 18 on 224 expressed items, the fact that it is the major concept expressed among other individual strategic benefits seems to indicate that having a prior knowledge helps reflecting about one's learning process. This reflective thinking – also present in other student items – is a good point in itself, as "*reflection is linked to elements that are fundamental to meaningful learning and cognitive development*" [22].

As for strategic *collective* benefits (5 items), students express "*catching up*", or "*leveling*" (4/5) and "*give basics to all students*" (1/5). Thus, some students are aware of their cohort's heterogeneity and maybe some perceive the usefulness of building a learning community. Unfortunately, current survey questions do not allow to dig further in this direction.

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 utility		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 exercising period		33.33%

Categories of 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 going to subcategories inside the category.

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%)**	
mutual aid		57.89%
hints contribution		42.11%
Technical aspects	**14 (11.66%)**	
tool usability		28.59%
access device		21.42%
availability		21.42%
remote access		21.42%
special characters		7.15%
Availibilty of lab rooms	**8 (6.66%)**	

Categories of facilitating factors reported by student answers. For each category, the number (and percentage over 120) of student items is indicated, together with the relative percentage going to subcategories inside the category.

The *Confidence* entry in Table 4 shows that the learning situation reassured students on four levels: self-confidence ("*I could gain self-confidence* ", "*I feel more independent/autonomous*"); on the "*skills*" or procedural knowledge they own in the end; on their conceptual knowledge ("*Now I know basic Unix commands*" / "*new commands*", "*I know how to use Unix now*"); the utility of their practical knowledge ("*widely used system in CS*").

Students also identify the *Assessment method* as a knowledge acquisition factor, pointing at the playful aspects of the tool (4/10 items), *i.e.*, getting hints in feedback, being able to improve their score ("*it is challenging*", "*I learned without having the impression to work, which is the most important*"). The fact that we used *scores* (from 0 to 100%) as in some games rather than grades also seem to motivate them (3/10), they like being able to retry and reach a better performance. This highly correlates with the learning analytics showing than > 85% (2020), resp. > 79% (2021), students improved the score of their first trial at an exercise.

Some students points at the learning pace efficiency (3/9 items) ("*allows to quickly learn basic commands*" with a "*a rather efficient learning rhythm for me as beginner*"). They highlight the time freedom (3/9) left to answer an exercise ("*we can work when we want*"). The duration of the one-week exercise part seems appropriate to some students (3/9) ("*Doing exercises over one week to anchor skills is a very good thing*").

4.4 Students identify facilitating factors

We identified 14 factors in student items that relate to assumption (4), *i.e.*, to the identification by students of factors influencing their learning. We grouped these factors in four categories (Table 5).

A most recurring factor (18/79) is the difficulty level of the exercises, expressed comments being mostly that exercises are too easy. In contrast, students highlight good points of the exercises, praising them for their progressive (11/79) and varied (6/79) nature. This highlights once more the heterogeneity among students in the cohort but also hints that exercises are more tailored to novices than knowledgeable students. Indeed, at this point in the OS course, our main goal is that *all* students get a *minimal* procedural knowledge at the end of this 2-week learning situation. Though, this indicates that we need to rework some of the current exercises to increase their challenging aspect for advanced students or to propose additional exercises targeting them. Items concerning assessment modalities welcome the possibility to do multiple trials at exercises and to obtain feedback on their answers (9/79). Students appreciate the assistance they get both from mutual aid possibilities (11/19) as well as hints given in statements and feedback (8/19). Technical aspects of the learning tool concern difficulties to choose a device to connect (3/14) or in contrast the ease of access (4/14).

5 DISCUSSION

We presented above both a new web-based exercising tool to foster the practical learning of the Unix system and an in-depth analysis of its use in a learning situation with four successive cohorts.

The tool offers a short list of auto-gradable exercises, each centered around specific procedural skills. When submitting their work, students get detailed feedback on their answers, including explanations on some of their errors, hints to improve and a score. Both these elements encourage them to enhance their proposals and to grow in skills. Instructors can set up sessions for their students on

our deployed instance of the tool, and propose new exercises. They can also deploy their own instance as the tool is freely distributed. Lastly, the tool can be integrated with LMS platforms.

The analysis of a learning situation relying on the tool teaches us that it actually allows students to improve their procedural knowledge. Though rudimentary, the playful aspects of the tool put them in a game-like situation motivating them to retry and then improve. The results also show that a group dynamics emerged after the two-weeks learning situation and that they are able to perceive a re-activation or a gain of procedural knowledge. The strategic benefits of the experience and the confidence they get from it are the most frequent feedback. They also indicate that exercise components (difficulty level, statement, timing, ...) were the predominant facilitating factors.

This feedback gives insight on the way students react to such tools and on their potential for hybrid or distance learning. Moreover, the learning situation and the survey that followed helped students develop metacognitive capabilities [9, 26], *i.e.*, their ability to think about their thinking and to self-evaluate, an important quality that will allow them to do better work in the future [22].

As a whole, these analyses lead us to consider the learning situation related here as an *enabling environment*, *i.e.*, a technical and social environment providing individuals 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 [11].

ACKNOWLEDGMENTS

This work was funded by *Fondation Polytech*, the *Polytech Network* in the IDEFI AVOSTTI ANR project. It was supported by the *Comité Numérique pour la Formation* of *Université Montpellier* and by the IT service of Polytech Montpellier, 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.katacoda.com/. Online; accessed January 2021.
[5] J. Biggs. 1996. Enhancing Teaching Through Constructive Alignment. Higher Education 32 (10 1996), 347–364. https://doi.org/10.1007/BF00138871
[6] S. 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
[7] G. Brousseau. 1988. Théorie des situations didactiques. La Pensée Sauvage, Grenoble.
[8] S.M. Doane, J.W. Pellegrino, and R.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
[9] Anastasia Efklides. 2001. Metacognitive Experiences in Problem Solving. Springer Netherlands, Dordrecht, 297–323. https://doi.org/10.1007/0-306-47676-2_16
[10] C. Estler and N. Nordio. [n.d.]. https://codeboard.io/. Online; accessed January 2018.
[11] P. Falzon. 2005. Ergonomics, knowledge development and the design of enabling environments. In HWWE.
[12] R.M. Hoar. 2014. Generally Educated In The 21st Century: The Importance Of Computer Literacy In An Undergraduate Curriculum. In WCCCE.
[13] T. Kendon and B. Stephenson. 2016. Unix Literacy for First-Year Computer Science Students. In Proceedings of the 21st Western Canadian Conference on Computing Education, WCCCE '16, Kamloops, BC, Canada, May 6-7, 2016, Surinder Dhanjal and Faheem Ahmed (Eds.). ACM. https://doi.org/10.1145/2910925.2910930
[14] R.B. Lewis and S.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
[15] R. Lobb and J. Harlow. 2016. Coderunner: a tool for assessing computer programming skills. Inroads 7, 1 (2016), 47–51.
[16] S. Mangul, L. S. Martin, A. Hoffmann, M. Pellegrini, and E. Eskin. 2017. Addressing the Digital Divide in Contemporary Biology: Lessons from Teaching UNIX. Trends Biotechnol 35, 10 (10 2017), 901–903.
[17] R. 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
[18] G. Molinari, B. Poellhuber, J. Heutte, E. Lavoué, D. Sutter Widmer, and P.-A. Caron. 2016. L'engagement et la persistance dans les dispositifs de formation en ligne : regards croisés. Distances et médiations des savoirs (online) 13 (2016). https://doi.org/10.4000/dms.1332
[19] M. 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. ACM Press. https://doi.org/10.1145/1999747.1999776
[20] Pluralsight. [n.d.]. https://www.pluralsight.com/codeschool. Online; accessed November 2021.
[21] B. Poellhuber, N.G. Roy, and I. Bouchoucha. 2019. Understanding Participant's Behaviour in Massively Open Online Courses. The International Review of Research in Open and Distributed Learning (2019).
[22] C. Rolheiser, N. C. Rolheiser-Bennett, B. Bower, and L. Stevahn. 2000. The portfolio organizer: Succeeding with portfolios in your classroom. ASCD.
[23] M. Suppa, O. Jariabka, A. Matejov, and M. 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, 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] K. von Hausswolff. 2017. Hands-on in Computer Programming Education. In Proceedings of the 2017 ACM Conference on International Computing Education Research (Tacoma, Washington, USA) (ICER '17). Association for Computing Machinery, New York, NY, USA, 279–280. https://doi.org/10.1145/3105726.3105735
[26] L.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] F. Wang and M.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