© [2006] IEEE. Reprinted, with permission, from [Andrew Solomon, Daniel Santamaria and Raymond Lister, Automated Testing of Unix Command-line and Scripting Skills, ITHET '06. 7th International Conference on Information Technology Based Higher Education and Training, 2006.]. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Technology, Sydney's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it

Automated Testing of Unix Command-line and Scripting Skills

Andrew Solomon, Daniel Santamaria and Raymond Lister

Abstract—**This paper describes the use of LinuxGym - software which assists a student's learning of Linux scripting, from directory listing to advanced Perl programming. The software randomly generates new tasks for each student and is used to check their attempts for the purpose of both formative and summative assessment. While the tasks are smaller than an assignment-sized software development project, they are nevertheless actual Linux usage (rather than a simulation) and enable a large number of practice and feedback loops, rather than just one or two large submissions. As a result, there has been a far lower failure rate and student surveys indicate a high level of motivation and satisfaction with its use.**

***Index Terms*—Computer science education, Operating systems**.

I. Introduction

NO matter what teachers list as the topics for study, students will study what they believe will be assessed. Furthermore, the students' approach to study will also be influenced by the medium through which we assess. Rowntree [7] observed the following about the relationship between assessment and the curriculum:

> *"If you wish to discover the truth about an educational system, we must look into its assessment procedures. What student qualities and achievements are actively valued and rewarded by the system? How are its purposes and intentions realized? To what extent are the hopes and ideals, aims and objectives professed by the system ever truly perceived, valued and striven for by those who make their way within it? The answers to such questions are to be found in what the system requires students to do in order to survive and prosper. The spirit and style of student assessment defines the de facto curriculum*." [page 1]

Related to Rowntree's observation is Biggs principal of Constructive Alignment [3]. Biggs observes that students construct meaning from what they do to learn, and he advocates that a teacher should align the learning outcomes, the learning activities, and the assessment tasks. Biggs describes the impact that the grading method has on the student's learning approach as the 'backwash effect'.

No matter how well the outcomes and activities are aligned, the student learning experience will not be worthwhile unless there is also good feedback. Ramsden has argued that the quality of the feedback is the major determinant of student performance. From data acquired via the Australian Course Experience Questionnaire, Ramsden found that the survey item that correlates best with student performance is "Teaching staff here normally give helpful feedback on how you are going" [6, p107].

Manual feedback, however, can be very demanding upon teachers. An assessment method must be practical. It must be achievable within the resources available. In teaching large classes, the most limited resource is often the time of the teachers.

Toohey [8] argues that giving students practical, professional tasks to perform for grading has 'clear relevance' to professional education. Ramsden cites Newble and Clarke as having established the principle that problem-based learning simulates the type of problems met in professional life and is 'more likely to encourage students to adopt a deep learning approach'.

A. Criteria for Evaluating Automatic Assessment

In the light of the above discussion, we defined the following three-fold objectives for teaching Unix skills:

a) to grade students using an approach that accurately determines their individual Unix command formulation skills;

b) to grade students in a manner that closely replicates the way that they will use their Unix skills in the real-world; and

c) to encourage students to practice and develop their Unix skills online.

All three authors are associated with the University of Technology, Sydney NSW 2007, Australia.

Andrew Solomon - phone: +61 2 9514 7938;

fax: +61 2 9514 4545; e-mail: andrews@it.uts.edu.au.

Daniel Santamaria (e-mail: dmsantam@gmail.com).

Raymond Lister (e-mail: raymond@it.uts.edu.au).

While designing the grading with Biggs' backwash effect in mind, we also considered Toohey's factors for selecting a grading method:

i) validity of the grading, which is how accurately it reflects the learning objectives for the subject;

ii) reliability of the grading, where a highly reliable method is one where work submitted for grading on different occasions should return similar results;

iii) how well the grading leads to and enables real learning. That is, as described by Biggs and Ramsden, the way that work is graded has an enormous influence on the approach that a student takes to learning in a subject.

No matter how much time the teacher puts into the providing feedback, that feedback is wasted unless students are practising their skills intensively. Anderson [1] observed the following:

> *"It requires at least 100 hours of learning and practice to acquire any significant cognitive skill to a reasonable degree of proficiency. For instance, after 100 hours a student learning to program a computer has achieved only a very modest facility in the skill."*

In the subject described in this paper, we expect students to spend 30-40 hours practicing their Unix skills. Therefore, by Anderson's definition, we do not expect students to become "proficient" in those 30-40 hours. In the paper-based approach that preceded the automatic system described in this paper, it was difficult to get students to practice their Unix skills for anything like 30-40 hours. Furthermore, if students had practiced for that length of time in the paper-based approach, it would have been impossible for teaching staff to provide quality feedback.

II. Previous Approaches to Unix skills teaching and grading.

There are a number of approaches to assessing Unix skills from which LinuxGym has evolved. In this section we review these approaches, including the approaches used at the author's institution prior to the introduction of LinuxGym. Unix skills were taught at both the undergraduate and postgraduate level.

A. Undergraduate subject: Command-Line Interface

In the undergraduate programme, the teaching focus is upon elementary unix functions such as cp, ls, rm, pipe, sort, chmod, and vim. Commonly used approaches for assessing students on such skills are multiple choice and short answer questions. Both approaches provide accurate summative assessment for the important skill of reading scripts before editing them. Both approaches are used in certification-type exams such as those of the Linux Professional Institute [5]. However this process is not at all formative - even when the answer is shown, the method by which it came about is not explained.

The converse assessment approach is to describe a task and have the student indicate which command (and parameters) will perform that task. In this case there is a great difference between multiple choice and short answer. There are many sequences of commands to perform the same task, and multiple choice questions can only provide one of those possibilities; perhaps a possibility not yet known by a competent student. On the other hand, if the student is permitted to write down the command they would use, then a correct assessment relies on the knowledge of the person marking the test. Again, even the most competent of Unix experts do not instantly recognize every valid way of performing a task. The Linux Professional Institute has attempted to automate this approach by having a Javascript simulation of a terminal. The simulation checks that the command written by the student is correct, however it does not permit much variation apart from the precise format of the command. In general, as these tests do not directly use Unix to determine the correctness of a student's answer, there is little incentive for students to practice the actual skills on a Unix machine.

Prior to the introduction of LinuxGym, the author's institution assessed students semi-manually. Students ran a program ("script") which saved the input and output of shell commands to a text file. In addition the students wrote an explanation of how their approach worked. Both the text file and the student explanation were then read, in hardcopy, by the marking staff. This had the benefit that, even if the marker was not familiar with the student's solution, the explanation could be assessed. Unfortunately, this manual approach to marking was time consuming. A marker could only process 50 student submissions per day. This bottleneck led to a students being limited to three iterations of the assignment task.

B. Postgraduate subject: Scripting with Bash and Perl

In the undergraduate programme, the teaching focus was upon bash and Perl scripting. Prior to the introduction of LinuxGym, the author's institution assessed the students via two large assignments. Each assignment required a student to write a single script containing a number of different Unix functions. The first assignment cannot be completed until at least half way through the semester because of the combination of skills required. By the time it was marked and returned to students, the second

assignment had been submitted. Thus students did not benefit from any feedback before attempting the second assignment. A high rate of plagiarism was suspected, and the overall failure rate was 30-50%.

III. LinuxGym

Beginning, 2002 the authors developed LinuxGym - software for automated marking of command-line operations and scripting. Since then it has been used to assess students in both the undergraduate and postgraduate courses described above. There were already unit testing systems such as BOSS [2] to automate the marking of programming languages. However, the effect of unix commands is frequently to alter files in some way (e.g. file permissions), not to produce output as with programming languages. Therefore, LinuxGym uses a different approach from program marking systems.

A description of a session with LinuxGym is now given, from the viewpoint of a student. When the student logs into the machine "unixexam" via ssh, their home directory will contain the file "instructions.txt", an example of which follows:

Question 1: There is a file in your home directory called 7sidd10.txt. Append the contents of /usr/local/unixexam-data/gutenberg/7myrt10.txt to the end of 7sidd10.txt.

Question 2: Write a perl script called perl_hubbywife.pl which takes one argument - a file containing a list of <husband>:<wife> partnerships where each woman may have more than one husband, but each man has at most one wife (see /usr/local/unixexam-data/newstuff/wifedata.txt as an example).

For each wife (in alphabetical order) the function prints her list of husbands, on the same line, in alphabetical order. Example:

```
[solomon]$ ./perl_hubbywife.pl  /usr/local/unixexam-data/newstuff/wifedata.txt
Samantha:Isaac+
Savannah:Ian+
Sophia:Brandon+Connor+David+Joshua+Matthew+Noah+
[solomon ]$
```

In Question 1, the names of the files are randomly selected from the set of all files in the directory /usr/local/unixexam-data/gutenberg/, and 7sidd10.txt. has been copied into their home directory. Now, the first thing the student might do is to experiment with the marking, which can be done as many times as they like:

```
[solomon]$ mark_question 1
QUESTION 1 ..... WRONG:( - Try again.
[solomon]$ cat /usr/local/unixexam-data/gutenberg/7myrt10.txt >> 7sidd10.txt
[solomon]$ mark_question 1
QUESTION 1 ..... RIGHT - Well done!
[solomon]$
```

In regard to Question 2, there is no randomness in the question itself, but if the student has written the script perl_hubbywife.pl the marking script will call it with different data-files and compare its output with the output of the perl_hubbywife.pl written by the teacher, marking it as correct if the outputs are the same for all data-files.

Finally, when the exam is over, the account is closed and the list of <question>:<mark> is stored in a file in the directory to which the teacher has access. Another file with a list of <student_id>:<total> is also stored and this is easily loaded into the spreadsheet of final marks for the test.

An actual test consists of 10 questions to be completed within an hour. For the undergraduate students there are three such tests in the semester, assessing their dexterity with Unix commands. Postgraduate students have seven tests, covering a range from tests similar to those for the undergraduates through to Perl scripting.

IV. Evaluation – Students

Housego and Freeman [4] point out that technology-supported teaching is effective only when based on teaching practices which motivate students to adopt a deep learning approach, not because information technology is used simply for its own sake. To verify that LinuxGym is effective, we have evaluated it using two structured questionnaires, end of semester in-class discussion, and an online discussion forum. The students are from the postgraduate subject which was previously assessed with two large assignments. In the semester in which the students completed the survey, they were given seven LinuxGym tests, two multiple choice theory tests, and a single assignment.

The first survey is a generic university questionnaire (Table 1) with an emphasis on overall subject objectives. The second questionnaire was more tightly focussed on LinuxGym as a software tool, and how it had impacted on student learning and assessment.

Table 1. Generic university questionnaire

	Statement
Q1	I found the assessment fair and reasonable.
Q2	My learning experiences in this subject were interesting and thought provoking.
Q3	There were appropriate resources available to support

	the subject.
Q4	I received constructive feedback when needed.
Q5	Overall I am satisfied with the quality of this subject..

For each question, students are required to select only one of the responses shown in Table 2. The table also shows the percentage of students indicating a particular response prior to the introduction of LinuxGym. Table 3 contains the same information from students after the introduction of LinuxGym.

Table 2. Student responses to questions in Table 1, before introduction of LinuxGym

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	Average Response
Q1	24	30	21	15	9	Neutral
Q2	20	43	11	11	14	Neutral
Q3	25	31	25	8	11	Agree
Q4	17	46	14	11	11	Neutral
Q5	14	31	22	17	17	Neutral

Table 3. Student responses to questions in Table 1, after introduction of LinuxGym (Spring 2005)

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	Average Response
Q1	21	54	19	6	0	Agree
Q2	25	44	23	8	0	Agree
Q3	23	60	13	4	0	Agree
Q4	15	54	27	4	0	Agree
Q5	25	51	17	6	2	Agree

The numbers for each question improved markedly once LinuxGym was deployed. One aspect in particular which was most encouraging was the reduction in the number of negative comments. Whilst the strongly agreeing student numbers only showed a small improvement, negative comments were vastly reduced. Beginning with 19%-34% of students providing negative feedback, this dropped to 4%-8% once LinuxGym was deployed.

Due to the feedback and decreased failure rate, LinuxGym appears to have its greatest positive effect on students who previously failed the subject. By providing a means to practice, and get feedback in real-time, previously less-able students are improving their learning, and are subsequently more satisfied with the delivery of the subject. Furthermore, the ability to demonstrate their skills in the real life environment provided by LinuxGym makes it easier to evaluate the knowledge of the student with increased accuracy.

Table 4 shows the questions asked in the second questionnaire, which directly focused on LinuxGym. The number of responses for both true and false options are also indicated in the table.

Table 4. Results of questionnaire focussed on LinuxGym in particular.

	Statement	True	False
Q1	LinuxGym is more motivating to practice Linux than a written test.	26	1
Q2	LinuxGym is more motivating to practice Linux than a written assignment.	26	1
Q3	LinuxGym helped me to improve my Linux skills.	26	1
Q4	I prefer LinuxGym to a written test.	19	8
Q5	I prefer LinuxGym to a written assignment..	19	8
Q6	LinuxGym has given me an accurate idea of my Linux skills.	25	2
Q7	LinuxGym marking was consistent and fair.	19	8

The majority of students indicated that LinuxGym motivated them to practice Linux more. This is also clearly reflected in the significantly higher pass rate in the semester where LinuxGym was employed. Past experience in teaching Linux has shown a correlation between practice and final grades.

Whilst students were willing to concede that LinuxGym both motivated practice and helped to improve their skills, there was still approximately 30% of respondents who preferred a written test. This may be attributed to the fact that a written test has a greater likelihood of accommodating small errors.

The exact same number of students (approx. 30%) also claimed that the marking was not consistent and fair. So whilst almost all students accepted that LinuxGym provided both motivation and improvement of their Unix skills, some are not entirely satisfied that they achieved the mark that they deserved. Anecdotal evidence suggests that those students may have felt that more time was required to complete the test which is not a direct criticism of LinuxGym.

In the three semesters prior to the implementation of

LinuxGym, the postgraduate failure rate varied between 30% and 50%. After deployment in Spring 2005, the failure rate dropped almost to zero. Through the use of LinuxGym, students were better able to gauge their skills before an exam, and hence were more prepared for assessments in terms of the level of knowledge required of them.

Qualitative Data

As another source of estimating student satisfaction, end of semester discussions took place within each tutorial, and an online forum was provided to allow students to express their thoughts on LinuxGym.

Many students commented that the ability to access LinuxGym and participate in practice questions allowed them to work at their own pace at a convenient time. Whilst a teacher may be able to answer questions, LinuxGym can allow students to gather together questions for the tutor. Upon receiving tutor feedback, LinuxGym is always available to verify and consolidate their understanding.

Another common theme was the realistic approach to teaching Unix. Previous criticisms of classes prior to the deployment of LinuxGym indicated that there was no need to really understand the application of Unix commands to pass a paper based exam. More often than not, a memorisation approach could be sufficient. One needs to evaluate a student's Unix skills by their interaction with Unix itself. LinuxGym requires that the student demonstrate their ability in a real world environment, receiving feedback as one would expect in the workplace. Students declared that the realistic nature of LinuxGym provided them with motivation to practice, and that it made the learning process more interactive and interesting.

The majority of students were pleased with the speed of the feedback. LinuxGym gives them on-going feedback throughout the exam (configurable), and instant results once the exam has been completed. Students were pleased to be able to learn of their marks whilst the exam was still fresh in their minds. Some also claimed that this made it easier to dispute any alleged discrepancies in grades.

V. EVALUATION – TEACHERS

In this section we present the teacher's perspective on the benefits of using LinuxGym, as well as suggestions for how it could improve.

One of the main benefits is the fact that it increases the amount of feedback the students receive while taking less marking time from the teachers. The students have a far better idea of what their problem is before they ask a question of the teacher, and the teacher has more time to spend addressing it – particularly helpful for students for whom English is a second language.

Randomly generated parameters of the questions enables the teachers to present the same type of question to the student many times so that the student manages to generalize the problems they are solving. Two particularly good examples are:

- write an awk script which takes a file of 10 lines of 10 comma separated numbers, and performs the sum of some randomly selected row or column; and
- create a file with some randomly specified set of permissions.

In both cases, the necessary code to model the human language description is quite complex – in the first case loops, and in the second case logic. The only effective way to attain the skill of translation from intuition to code is practice.

The software also provides some assurance of quality. In terms of plagiarism, the randomness of the question parameters ensures that it is generally not sufficient to get a correct answer simply by copying another student's solution. In terms of a "weak solution" for scripting which is only partly correct, a well written question will be able to identify its flaws by a process of multiple randomised testing inputs.

Overall, the greatest pleasure for the teachers was the fact that for a class of 80 masters students who for several years had had a 30-50% failure rate, almost everyone now passes the subject.

Nevertheless there are some obvious improvements to LinuxGym from which the teachers would benefit. The first is that it is quite arduous writing questions – a Perl script of (on average) about 85 lines. It would be preferable to have a GUI interface by which the question and its testing could be defined at a high level and the associated Perl script developed automatically.

Finally, it would be useful to have an easily accessible statistical analysis of the number of attempts a students has made for each question. From this one could identify which questions students find very difficult. Under some circumstances, this indicates a poorly written question, and in others it pinpoints gaps in the lecture content. Conversely, instead of binary marking – it would be nice to give students more marks if they get a question right with fewer attempts.

VI. CONCLUSION

An online examination system for Unix commands and scripting was introduced and the results of its evaluation have been described. Our goal was to provide a system that aligned the grading strategy with the way students will use Unix after graduation in order to encourage deep learning. The results of the students' evaluation indicate that our

system achieves that goal.

REFERENCES

[1] Anderson, J. R. Acquisition of cognitive skill. Psychological Review, 89:369–403, 1982.

[2] BOSS Online Submission System, Available Online at: http://www.dcs.warwick.ac.uk/boss/info.html, [accessed March 7, 2006]

[3] Biggs, J. *Teaching for quality learning at University*. Buckingham, Open University Press,1999.

[4] Housego, S. and Freeman, M. Case studies: integrating the use of web-based learning systems into student learning. *Australian Journal of Educational Technology*, 16(3), (2000), p258-282.

[5] Linux Professional Institute, Available Online at: http://www.lpi.org [accessed March 6, 2006]

[6] Ramsden, P. *Learning to teach in higher education.* London, Routledge, 1992.

[7] Rowntree, D. (1987). Assessing Students: How shall we know them? London: Kogan Page.

[8] Toohey, S. *Designing courses for Higher Education.* Buckingham, Open University Press, 1999.