1983

A tutorial introduction to the submit system

Cecily S. Bailes
University of Wollongong

THE UNIVERSITY OF WOLLONGONG

DEPARTMENT OF COMPUTING SCIENCE

A TUTORIAL INTRODUCTION TO THE SUBMIT SYSTEM

Cecily S. Balles

Department of Computing Science
University of Wollongong

Abstract

An automated student programming assignment submission system is presented. Authorised tutors can define submission parameters for particular assignments, including the setting of deadlines and the testing of submitted programs. Students can submit documentation, program source and test data files, and authorised tutors can then access the collected results of the students' submissions for assessment.

Preprint No 83-7 **July 26, 1983**

P.O. Box 1144, WOLLONGONG, N.S.W. AUSTRALIA
telephone (042)-282-981
telex AA29022

A Tutorial Introduction to the Submit System

Cecily S. Bailes

Department of Computing Science
University of Wollongong

INTRODUCTION.

The Submit system is a collection of UNIX* data files and programs that administers student assignment submissions. It provides the following commands:

(i) **labset**, used by a *tutor* to define submission parameters for a particular assignment in a particular subject, and by a *student* to obtain selected information regarding the parameters so defined

(ii) **submit**, used by a *student* for submission of an assignment

(iii) **verify** used by a *student* to check the results of a submission that was produced by **submit** or by a *tutor* to obtain the results of students' submissions for scrutiny as part of the assessment process

(iv) **delete** used by a *tutor* to remove submissions from the system.

A *tutor* for a given course is identified by the appearance of his user-id in the authorised tutors file, /pub/submit/admin/tutor. A *student* is identified by an entry in the student database /pub/students/enrolments. The submission parameters for an assignment include the deadline for the assignment, the compulsory EXPLAIN lessons which must be completed before a submission is accepted by the system and information regarding the processing of the student's assignment submissions.

The original Submit system was written by Charles M. Francis of the University of Wollongong in 1981. It was enhanced by Cecily Bailes in 1982. Throughout this document, when interactive dialogues are being described, computer type-out is displayed in **bold** type, and user response in ordinary type.

LABSET - DEFINITION OF SUBMISSION PARAMETERS.

The command

labset -i

is used by a *tutor* to define an assignment scheme to the system. The submission scheme parameters are established via an interactive dialogue. Initially, the prompt

Course name:

is given, to which the *tutor* responds with the three-digit code number of the relevant subject, optionally preceded by the mnemonic letters 'cs'. For example

Course name: 101

defines CSCI101 as the subject of interest, as does

Course name: cs101

whereas

Course name: 945

does so for subject CSCI945 (Note: invalid course numbers are rejected). Then follows the prompt

*UNIX is a Trademark of Bell Laboratories.

Assignment number:

to which the *tutor* responds with the number of the assignment of interest. For example

Assignment number: 1

specifies assignment number 1. Submission schemes for assignments need not be created in an ordered sequence, a submission scheme for assignment 10 can be created before one is created for assignment 2.

The next prompt

Any prerequisite lessons (y/n)?

inquires whether or not successful completion of any exercises from the **explain**[1] system is a pre-requisite for submission of the now-determined assignment. A response of 'n' indicates a negative reply, whereas one of 'y' indicates a positive reply, generating the production of the prompts

Enter list of lessons
>

after which the *tutor* enters, line-by-line, the names of the pre-requisite lessons until an empty response (just press the RETURN key) is given. For example,

Any prerequisite lessons(y/n)? y
> assignment
> while-loop
>

specifies that **explain** lessons 'assignment' and 'while-loop' are pre-requisites for the current assignment. Note that **labset** checks for the existence of each lesson (by the presence of the relevant file) and rejects any non-existent lesson with the following comment:

non-existent lesson

Having dealt with the issue of pre-requisite lessons, the prompt

Documentation shell:

seeks the name of a file, which is an executable shell program or procedure. This program will process the submitted student documentation files. Similarly, the succeeding prompt

Program shell:

seeks the name of a shell program file to process the submitted student program source files, followed by

Data shell:

seeking the name of a shell program file to process the submitted student test data files. In each case, an empty response signifies that the relevant part of the student's submission is to be discarded.

As will be seen in the discussion of the **submit** program, the shell programs are executed with the names of student's files as arguments, and the standard output of these executions is redirected, by **submit**, into a file named in the form

/usr/spool/submit/csnnn/l.nn.uid

where *csnnn* is the code number of the course (e.g. cs101), *l* is the laboratory group (e.g. A) of the submitting student, *nn* is a two digit representation of the assignment number (e.g. '04') and *uid* identifies the student.

As an example, if the documentation, program and data shell files are the standard shell files *doc1*, *prog1* and *data1* respectively and the content of *doc1* is

```
IFS=" \t\n"
PATH=/bin:/usr/bin
cat $*
```

and *prog1* contains

```
IFS=" \t\n"
PATH=/bin:/usr/bin
if test $# != 1
then
        echo; echo; echo;
        echo "Too many program files, only one please."
        echo; echo; echo;
        exit
else
        pi -l $1
        mv obj a.obj
fi
```

and *data1* contains

```
IFS=" \t\n"
PATH=/bin:/usr/bin
if test ! -r a.obj
then
        echo; echo; echo
        echo "Program has not been successfully compiled."
        echo; echo; echo
        exit
fi
px a.obj <$1
rm a.obj
```

then the user is required to submit at least one documentation file, exactly one Pascal source file and a data file, e.g.

```
submit -c100 -a1 doc1.doc doc2.doc prog.p data.dat
```

The submission results will consist of the concatenation of *doc1.doc* and *doc2.doc* followed by the Pascal compiler listing of the program *prog.p* and the output of the execution of this program against the user input *data.dat* (if the compilation was successful).

The supplied shell files must be readable and executable by all so that students may execute these shell command files. They must reside in the directory /pub/submit/shell. If the shell command file (given in response to one of the above prompts) does not exist in the directory /pub/submit/shell, it will be copied into this directory and re-named in the form

Docu*xxx.nn*
Prog*xxx.nn*
Data*xxx.nn*

for each of the three categories, where *xxx* identifies the subject by its three digit number, and *nn* the assignment number.

The prompt

Section:

follows, inquiring for which part of the class (i.e. lab section or group) a specification of a submission deadline is to be given. The response can be either a letter identifying a lab group, or an asterisk '*' to signify all the lab groups for which no particular arrangements have been made. Then follows the prompt

Due date:

to which the reply is the chosen deadline for the assignment for the just-defined lab group. The deadline is entered in the **date** (1) format i.e. *MMddhhmm* where *MM* is a digit pair identifying the month (e.g. '05' means May), *dd* a digit pair giving the day of the month, *hh* the hour of the day, and *mm* the minute of the hour. The information is then re-presented in decoded form, so that we have for example

Section: f
Due date: 06181630
Fri Jun 18 16:30:00 1982

which establishes a deadline for group F at 4.30 p.m. on Friday 18/6/82. This dialogue is repeated until an empty response is given to a 'Section: ' prompt. For example

Section: *
Due date: 1119163000
Fri Nov 19 16:30:00 1982
Section: a
Due date: 10291700
Fri Oct 29 17:00:00 1982
Section: f
Due date: 10291700
Fri Oct 29 17:00:00 1982
Section:

specifies a due date on November 19 for the en
both due on October 29. Note the necessity to repeat the identical entry for groups A and F. The response to a "Due Date: " prompt must be in proper **date** (1) format, otherwise the following message is displayed:

Bad conversion--reenter:

The prompt

Delete extensions (y/n)?

determines whether the previous extensions given to certain individuals (if any) are to be retained or not. See Section on LABSET MISCELLANY with regards to setting of extensions for particular students.

The dialogue for establishing submission parameters is now complete, and execution of the labset program terminates.

LABSET - OBTAINING INFORMATION.

The command

labset

is used by a *student* to print a list of the due dates of an assignment. The dialogue proceeds as follows, for example

Course name: 222
Assignment number: 4

Course: cs222 **Assignment #04**

Lab Section	Due Date
A	Mon May 24 17:00:00 1982
B	Tue May 25 17:00:00 1982

displaying that the due dates for sections A and B of CSCI222 are 24/5/82 and 25/5/82 respectively at 5 p.m.

LABSET MISCELLANY

It is possible to signify the class and assignment number on the **labset** command line itself by the **-c** and **-a** options. For example

```
labset -c313 -a2
```

specifies subject CSCI313 and assignment number 2, avoiding the need for the initial interactive dialogue. Incidentally, invocation of **labset** by a *tutor* without a **-i** option first obtains information regarding subject and assignment number (in either of the described ways) and if no relevant submission parameters have been established, assumes a **-i** option, i.e. proceeds to determine these parameters by way of the interactive dialogue as described above. Otherwise, if the submission record exists, the *tutor* is presented with a list of assignment due dates, the names of compulsory EXPLAIN lessons and the names of the shell files used to submit an assignment. The **-p** option exists to enforce this behaviour, so that

```
labset -ccs313 -a3 -i -p
```

first establishes parameters for CSCI313 assignment 3, and then prints out a list of deadlines so established. The **-x** option is used to give extensions to individual students. The command

```
labset -c201 -x
```

first produces an enquiry as to the CSCI201 assignment number under consideration, with the usual prompt

Assignment number: 3

The user has specified assignment 3 and after the heading

Give individual extensions

and the prompt

User-id:

the user replies with the UNIX user-id of the student to whom the extension is to be given. This is followed by the prompt

Due date:

to which the user replies with the new deadline for the nominated individual. This procedure is repeated until an empty response to the **User-id** prompt is given, for example

Give individual extensions
User-id: jmf
Due date: 10101630
Sun Oct 10 16:30:00 1982
User-id: rjh
Due date: 10121700
Tue Oct 12 17:00:00 1982
User-id:

gives an extension to October 10 at 4.30 p.m. to jmf and one to October 12 at 5 p.m. to rjh. If the response to the prompt

Due date:

is non-numeric then the extension for that individual is removed. The **-d** option may be used to change deadlines on a lab group basis. The dialogue

```
$ labset -c211 -a4 -d
Set lab due dates
Section: a
Was due: Mon May 24 22:00:00 1982
Due date: 11060830
        Sat Nov  6 08:30:00 1982
Section: *
Due date: 11082200
        Mon Nov  8 22:00:00 1982
Section:
$
```

resets the deadline for CSCI201 assignment 4 for the lab group * to 8/11/82 and for group A to 6/11/82. Previously determined extensions are unaffected. For example, if an explicit deadline for group B has been established, the above dialogue will not alter it. At any time, Group * refers only to the groups for which to date no parameters have been explicitly given.

SUBMIT

The **submit** program is used to submit student documentation, program source, and test data files for processing by the procedures determined by **labset**. The command

```
submit -c334 -a10 file1 ... filen
```

submits file1 ... filen as the solution to CSCI334 assignment number 10. If the command is entered without options as in

```
submit file1 ... filen
```

then the subject and assignment numbers would be determined by the usual dialogue.

The distinction between documentation, program source, and test data files is made on the basis of the suffix of the name of each file. A suffix is defined as the character string following the right-most '.' (period) character in a file name. Documentation files should have suffices **.doc** and data files should have **.dat** or **.data**. Program source files should have suffices **.p .c .f .y** or **.bas**, informally signifying Pascal, C, FORTRAN, Yacc or BASIC language source programs. The order of the file names is maintained for each category, for example, the command

```
submit -c331 -a5 doc1.doc main.c test.data sub.c doc2.doc
```

causes the file doc1.doc followed by doc2.doc to be passed as arguments to the documentation shell procedure as determined by **labset**. The file test.data is passed to the data shell procedure, and the files main.c and sub.c become arguments one and two to the program shell procedure. If the relevant procedure does not exist, the respective submitted files are discarded. The results of the execution of each of the above shell procedures (in the sequence: documentation shell; program shell; and data shell) are concatenated into a unique file, identified as follows:

l.nn.uid

where *l* is the character identifying the lab group, *nn* is the 2 digit assignment number and *uid* is UNIX user-id of the person submitting the assignment. This file is placed in the directory

/usr/spool/submit/cs*nnn*

where *nnn* is the 3 digit course number.

VERIFY

The command

```
verify -c222 -a4
```

may be used by a *student* to check the submission of assignment 4 for CSCI222. Note

that the command

```
verify
```

would as usual first institute a dialogue to determine subject and assignment numbers. *Verify* then presents the result of the submission as retained by **submit** in the directory /usr/spool/submit.

The command

```
verify -c222 -a4
```

when executed by a CSCI222 *tutor* on the other hand displays on standard output the results of each student submission for assignment 4. The command

```
verify -c222 -a4 -p
```

sends the information to a printer, just as would

```
verify -c222 -a4 | opr
```

The command

```
verify -c222 -a4 -p9
```

specifies printer 9. The *tutor* may also use the **-l** option to specify a particular lab group, and the **-u** option to select a particular individual for example

```
verify -c333 -a2 -ujmf
```

displays the results of jmf's CSCI333 assignment 2 submission. Alternatively

```
verify -c311 -a7 -la
```

displays results of CSCI311 assignment 7 submissions for lab group A. The **-s** option can be used by a **tutor** to list the names of the stored student submission files for the nominated assignment. For example

```
verify -c311 -a7 -s
```

performs the same operation as

```
ls -l /usr/spool/submit/cs311/*.07.*
```

DELETE

The **delete** program allows a *tutor* to remove the submission result files (as indicated by the **-s** option to **verify**) in an interactive manner. For example

```
delete -c931 -a4
```

applies the operation

```
rm -i /usr/spool/submit/cs931/*.04.*
```

removing the results of assignment 4 submissions of CSCI931. As with **verify**, **-l** and **-u** options exist to selectively delete lab group or individual student results.

ACKNOWLEDGEMENT

My thanks go to Professor Juris Reinfelds for his comments on drafts of this document.

REFERENCES

[1] UNIX Programmer's Manual, Seventh Edition, Volume 1, The University of Wollongong.

APPENDIX

The relevant pages in Volume 1 of the UNIX Programmer's Manual are included here for easy reference.

NAME

submit, verify, delete, labset - student assignment submission

SYNOPSIS

submit [-cn] [-an] file.suffix [file2.suffix ...]
verify [-cn] [-an] [-lc] [-uname] [-pc] [-s]
delete [-cn] [-an] [-lc] [-uname]
labset [-cn] [-an] [-p] [-l] [-d] [-x]

DESCRIPTION

These programs together make up a scheme for submission of student assignments allowing automatic verification of programs, and easier administration. *submit* is used by students to actually submit the various programs/files that they have prepared. The assignment number must be specified, either with the -an option where n is the number, or by response to a prompt. The student's record will be looked up in the student enrolments file, and the course, if specified, must agree with the record. The course need only be specified if the student is enrolled in more than one course. It may be specified as an option, for example -c301 or -ccs301 or the program will prompt if necessary. The time will then be checked against the deadline for the assignment, and if this is passed, *submit* will check for any individual extensions. If the assignment is overdue, and the user does not have an extension the submission will not be accepted.

submit then groups the file arguments into three types of files based on their suffix: documentation files (ending in **.doc**), program files (ending in **.p**, **.c**, **.f**, **.y**, or **.bas**), and data files (ending in **.dat** or **.data**). Student files must follow these naming conventions if they are to be properly submitted (files without valid suffixes will be ignored). Each set of arguments will then be passed to a *sh*(1) command file specific to each assignment. Some assignments may require just documention, or just programs, while others may require both plus user-supplied test data. If any category of files is required, but not supplied, the submission will stop, and will indicate which files are missing. The output from all of the shell files will be concatenated onto one output file, constituting the student's submission. A student may resubmit as many times as he wishes, and each submission will overwrite the previous one, though a counter will be kept of the number of attempts.

The *verify* command is used to check the results of a submission. Both the course number and the assignment number must be specified, (if they are not given in the options, the program will prompt for them). If the user is not an authorized tutor for the course, only his own submissions (if any) will be displayed. Tutors will receive the submissions for the whole class, unless the lab section or user name is specified (using -lc where c is the one letter name of the lab section, or -uname where name is the login name of the student). In the latter case, all of the submissions for the specified lab section, or student, will be verified. If the -p option is specified, the submissions will be printed directly on the printer instead of on the standard output (the default). A default printer is used; the form -pc can be used to force output to printer *c*. (See *opr*(1)). (This form is preferable to redirecting the standard output to the printer, as each separate submission will be a separate job and start on a new page.) The -s option may be used to find out what submissions have been made using *ls*(1) with the -l option on each of the relevant files. The *delete* command is only effective for tutors, and will interactively delete the specified submissions, using the -i option on **rm**(1). If invoked by a student, *delete* will behave as *verify*, and print the submission on the screen.

labset is used to set up the due dates for each assignment. If invoked by a student, this will display the due dates for each section of the specified course, and list the names and dates for any students with extensions. Only authorized tutors may change specifications and due dates for an assignment. A number of options may be

specified to indicate what will be changed.

-i Initialize the submission scheme. The program will prompt for the names of the shell files to be used to process the student's submissions. Three files will be asked for which will receive as arguments the documentation files, program files, and data files submitted by the students. If any of these are not required, then a null line should be entered. The shell files should all be stored in the directory */usr/spool/submit/shell* which must be accessible to students, and each shell file should be executable by the students. This is because the user id of *submit* is changed to that of the student (it must be initially setuid(root) to allow restrictions on access to the submissions). Obviously, for security reasons, neither the directory nor the files should be writable by students.

-d Set lab due dates (this is implied by -i as well). *labset* will prompt for the due dates for each individual lab section. An '*' indicates all lab sections not otherwise specified. Only letters will be accepted as lab sections, and they are all currently mapped into upper case to match the enrolments file. Dates must be entered in the format of *date*(1). If there was already a deadline on the lab section, the old due date will be printed first. After the date has been entered, it will be reprinted in *ctime*(3) format for verification. If 'delete' is entered for the date (or any other alphabetic characters), the particular entry will be deleted. A newline terminates entry.

-x Give individual user extensions. *labset* will prompt for student user names, check that they exist, and ask for the appropriate due dates, in the same format as above. A null line for student name will terminate entry.

-p Print out the results after the submission has been described.

If no options are specified, then the default will be -i if no scheme for the particular lab as been set up, and -p otherwise (or if the user is not a tutor).

FILES

/pub/students/enrolments	student "data base"
/usr/spool/submit/cs*	submission spooling directories
/pub/submit/admin/tutors	file of authorized tutors for each course
/pub/submit/admin/labs	file of assignment descriptions and due dates
/pub/submit/admin/tmplabs	temp file for changing `labs'.
/pub/submit/shell	directory of submission shells

DIAGNOSTICS

If a submission is too late, an appropriate message will be printed. If an unauthorized access is attempted with *verify* or *delete* verification of the user's own submissions will be substituted instead. If no submissions have been made, *cat*(1) or *opr*(1) will complain that it cannot access the file.

labset will just print the details of the specified course assignment if the user is not an authorized tutor.

AUTHOR

Charles M. Francis,
University of Wollongong.