

**31473 – DATA STRUCTURE AND PROCEDURAL PROGRAMMING
SPRING 2007**

ASSIGNMENT 2 - DUE 11:59pm, 2/11/2007

This assignment is worth 25% of the total marks for this subject.

Requirement

For this assignment, you will write a C program that simulates the re-assembly of messages received from a packet-switching network.

Background

A data transmission system typically breaks a large message into smaller packets. In some systems, the packets can arrive at the destination computer out of order, so that before the application requesting the data can process it, the communications program must ensure that the message is assembled in its proper form.

For this program, the data packets will be read from a file. Each packet will contain the following fields, separated by colons:

- Message ID number. This will always contain either 3 digits or the word END.
- Packet sequence number. This will consist of exactly three digits.
- Text. No text line will exceed ninety characters. The text may contain any characters, including colons.

Some examples of packets are:

```
101:021:"Greetings, earthlings, I have come to rule you!"
232:013:Hello, Mother, I can't talk right now,
101:017:Here is a message from an important visitor:
232:015:I am being harangued by a little green thingy.
END
```

All packets with a particular ID number belong to a single message. Sequence numbers may not be contiguous, but the correct ordering for a message will always be in ascending order of sequence numbers.

To simplify the task, it can be assumed:

- All messages will be complete: each message will have at least one data packet.
- Each packet will have a unique sequence number: a sequence number will not be used again for a subsequent packet.
- The END packet will always be the last packet received: this means you do not have to know the number of packets in a message, or wait for pending packets once an END packet has arrived.
- All message IDs and sequence numbers will be correct: you do not have to check messages for valid characters, etc.

- ❑ Packets will be contained in a file. Each line of the file will contain one packet, but as indicated above, the packets will be in random order (except for the END packet). There will be no blank lines.

Program Action

1. Your program will get the name of the file containing the packets as a command line argument.
2. It will then read the file and build up all the messages in a linked list of linked lists using the `fgets` function. For each message ID, you should maintain a linked list of packets received, and each new packet for the message should be inserted into the linked list in the correct place, so the list is always correctly sorted.
3. When the END packet is received, all the messages should be printed, one packet to a line, in correct order, with a header giving the 3 digit message ID. The message ID should be padded with 0's if it is numerically less than 100. A blank line will separate out each message ID.
4. As a final step, the program should then correctly free up all the memory used in the linked lists.

For example, the output of the message above would look like

```
Message 101
Here is a message from an important visitor:
"Greetings, earthlings, I have come to rule you!"
```

```
Message 232
Hello, Mother, I can't talk right now,
I am being harangued by a little green thingy.
```

Details

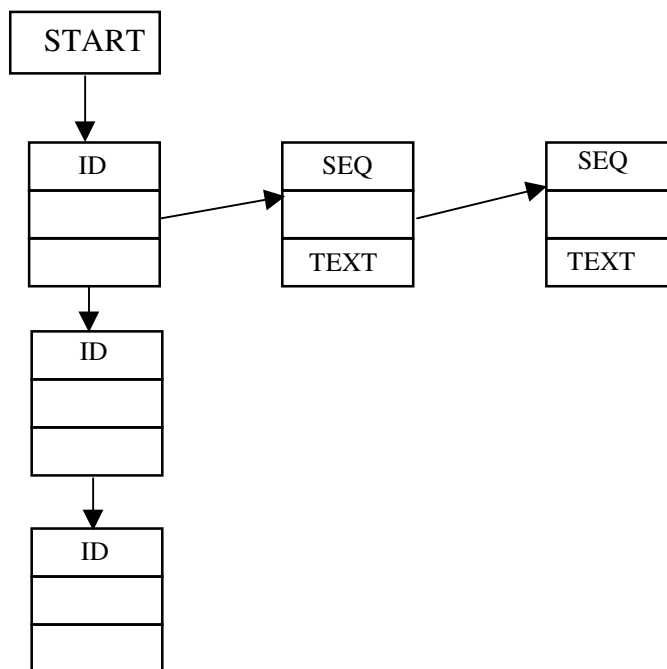
While there are other data structures that would work, you must implement a “linked list of linked lists”.

The base linked list contains an element for each message; consisting of a pointer to the next message ID, and a pointer to the first packet in the message. New IDs will be inserted into the list in numerical order.

Each message consists of a linked list of packets; where each list element consists of a pointer to the next sequence number and a line of text for that packet.

Whenever you need a new element, you should obtain the required space from dynamic memory using the `malloc` family, as discussed in lecture 8.

A diagram of the data structure follows



Benchmark Program

To clarify how the program works, a benchmark executable version has been placed on the server. You can run it by typing the following command

```
/home/glingard/packet message_file
```

Where `message_file` is the name of a file containing data packets. To give you an idea of the size of this assignment, the source code for the benchmark program, including whitespace and comments, takes about 280 lines of code.

Assignment Objectives

The purpose of this assignment is to demonstrate competence in the following skills.

- Program design
- Command line arguments
- Handling files
- Manipulation of linked lists

These tasks reflect all the subject objectives apart from objective 4.

As part of your subject workload assessment, it is estimated this assignment will take 22 hours to complete.

Queries

If you have a question, please contact the subject coordinator as soon as possible.

Gordon Lingard
 glingard@it.uts.edu.au
 9514-7935
 Room 04.559, Building 10

However, for frequently asked questions a FAQ file will be put up. Please check this file before emailing the coordinator with a question.

```
/pub/prprog/assign/assign2/faq.txt
```

There is already an assignment FAQ in the DS & PP web site. You should read this.

If serious problems are discovered the class will be informed and an update will be included in the following file

```
/pub/prprog/assign/assign2/errata.txt
```

Please keep a look out for this file.

PLEASE NOTE. If the answer to your questions can be found directly in any of the following

- subject outline
- assignment specification
- faq.txt
- errata.txt
- UTS Online discussion board
- DS & PP web page

You will be directed to these locations rather than given a direct answer.

PLEASE NOTE. Please do not send email in HTML format or with attachments. They will not be read or opened. Only emails sent in plain text format will be read. Emails without subject lines will be automatically deleted by the junk mail filters I have in place.

Assignment Submission

You will submit your program via the `submitass2` program. It is assumed that your C file is called `assign2.c` (although you may call it something else).

You must start in a directory that is within your home directory on `charlie` or `sally` and then run

```
/home/glingard/submitass2 assign2.c
```

This assumes that `assign2.c` is in the current directory.

The `submitass2` program will then perform a number of tests. These include.

Compiling

Your program must compile without fatal errors or warnings – using both the `cc` and the `gcc` compilers. Your program will be compiled with the following commands and options

```
cc  
gcc -pedantic-errors -Wmissing-declarations -Wall -Werror -ansi
```

You can learn what these compiler options mean by running `man cc` and `man gcc`.

PLEASE NOTE. Your program must compile on the student UNIX server. Programs written on Window's machines sometimes don't compile or run properly on the student server.

Style Feedback Program

Your program will be run through a style feedback program, which will check that your code meets a minimum style layout. If your program does not meet the standard a warning message will be printed. The style feedback program will display messages showing which lines of code do not conform to the standard and why they don't.

Code Feedback Program

Your program will be run through a code feedback program, which will check that your code meets minimum coding practices. If your program does not meet the standard a warning message will be printed. The code feedback program will display messages showing which lines of code are not acceptable and why they aren't.

Test Filter

Your program will be tested with 5 different sets of tests. The results of your program will be compared to the results generated using the benchmark `packet` program.

PLEASE NOTE. Make sure the output from your program is **EXACTLY** the same as that from the benchmark executable. **ANY** deviation of the output from your program to that of the benchmark executable will cause the test to fail.

Plagiarism Agreement

You will be required to agree to a statement that the assignment is your own work and that you haven't given your code to anyone else.

If all goes well, `submitass2` will reply with a message saying you have successfully submitted the assignment. It will also place in your current directory a file called `receipt.txt`.

You may submit your assignment as many times as you like. The last assignment received will be the one marked.

PLEASE NOTE. Only assignments submitted via the `submitass2` program will be accepted for marking.

receipt.txt

`receipt.txt` is your proof that you have submitted your assignment. Once you have received it, copy it to somewhere safe such as your home directory. Additionally, copy your C file into the same location. Do not modify them in any way. If you wish to continue developing your program then do it on a duplicate file.

The `receipt.txt` file contains three pieces of information

1. A line saying you have submitted your file and when you did it.
2. A checksum of your C file
3. A checksum of your receipt

If you tamper with your C file or the receipt then the checksums will become invalid and therefore your receipt will become invalid. No actions will be taken if receipts have invalid checksums.

Acceptable Practice vs Academic Malpractice

- Students should be aware that there is no group work within this subject. All work must be individual. However, it is considered acceptable practice to adapt code examples found in the lecture notes, labs and the text book for the assignment. Code adapted from any other source, particularly the Internet and other student assignments, will be considered

academic malpractice. The point of the assignment is to demonstrate your understanding of the subject material covered. It's not about being able to find solutions on the Internet. You should also note that assignment submissions will be checked using software that detects similarities between students programs.

- ❑ Do not let anyone submit their assignment from your account. The `submitass2` program copies your assignment into a secure directory based upon your user login name. Anyone else using your account will have their assignment placed in your directory. Students who do this will find themselves reported to the Faculty for possible academic malpractice and misuse of Faculty resources.
- ❑ Students are reminded of the principles laid down in the “Statement of Good Practice and Ethics in Informal Assessment” in the Faculty Handbook. Assignments in this subject should be your own original work. Any collaboration with another participant should be limited to those matters described in the “Acceptable Behaviour” section. Any infringement by a participant will be considered a breach of discipline and will be dealt with in accordance with the Rules and By-Laws the University. The Faculty penalty for serial misconduct of this nature is zero marks for the subject. For more information, see http://wiki.it.uts.edu.au/start/Academic_Integrity

Assignment Security

It is important to note that you have a responsibility to maintain the security of your assignment files. You can read more details about this in the Academic Malpractice section of the DS & PP web site - <http://learn.it.uts.edu.au/prprog/>

Assessment

Marks will be awarded out of 20 based upon the following scheme.

- ❑ Between 0 and 8 marks will be awarded by the computer based on the number of tests passed. The first three tests are worth 2 marks and the other two are worth 1 mark.
PLEASE NOTE. Some students have been known to write their code to artificially pass the tests rather than solve the assignment problem. In such cases a reduced mark will be given for the tests, including being given a 0.

The following marks will only be awarded if a score of 5 or more is awarded for the tests and both the code feedback and the style feedback programs are passed without generating warnings.

- ❑ Between 1 and 9 marks will be awarded on the quality of your code design and the algorithms used. The appendix in the lecture notes has a section called *Program Design* for further clarification on this.
- ❑ Between 1 and 3 marks will be awarded on the presentation style of the program. This involves meaningful variable names, intelligent use of comments and so forth. The appendix in the lecture notes has a section called *C With Style* for further clarification on this.

PLEASE NOTE. It is a fundamental requirement of this assignment that you use a linked list of linked lists to store packets, as outlined in the details section. Students who use any other data structure, including arrays, will receive 0 for the assignment.

For further details of the marking scheme please check out the assignments section of the DS & PP web page. <http://learn.it.uts.edu.au/prprog/>

Late Assignments, Extensions and Special Consideration

Please read the subject outline regarding late assignments and extensions. If you did not get the outline a copy can be found at the DS & PP web site

<http://learn.it.uts.edu.au/prprog/>

Assignments that are late by less than one week will incur a penalty of 1 mark for each day or part thereof late. Assignments more than a week late will not be accepted under any circumstance as the assignment solution will have been made available by then.

An extension of one week will only be granted if there is a fully documented reason which merits it. The documentation must be presented to the Subject Coordinator before the assignment due date. Extensions longer than a week will not be granted under any circumstance. If a one week extension is granted that means the assignment will be accepted up to a week after the due date without penalty. It will not be accepted later than that.

Students may apply for special consideration if they consider that illness or misadventure has adversely affected their performance in the assignment. For more information go to

http://www.sau.uts.edu.au/exams_ass/spec_cons.html

Getting Marks

You can check on your marks by running the following program.

```
/home/glingard/getmarks
```

Apart from your marks this program will give the following information.

- Given out – the date the assignment is given out.
- Due date – the date the assignment is due.
- Late date – assignments will be accepted up to one week after the due date but with a penalty of –1 mark per day or part thereof late.
- Marks released – The date the marks are released.
- Close date – The assignment is closed and the solution will be released. The close date will be one week after the assignments are marked and given back.

PLEASE NOTE. It is your responsibility to check that I have received your assignment and given you a mark. Even if you have a receipt you should check your mark and inform me if there is any problem before the close date. The `getmarks` program allows you to do this very easily. Unless you have a valid receipt or there are exceptional circumstances (e.g. serious medical conditions) no further correspondence regarding the assignment will be entered into after the close date.