Department of Computer Science

COS284 Practical and Assignment 2: System Calls

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

This document contains both practical 2 and assignment 2. In general the assignments will build upon the work of the current practical.

1.1 Submission

The Practical will be marked by the tutors in the practical sessions on 13 and 14 August, as well as fitchfork, and must be uploaded during or before the practical sessions.

The Assignment will be fitchfork only and is due at 17:00 on Friday 17 August. You may use the practical sessions to work on your assignment in the labs and ask for assistance if it is required.

1.2 Plagiarism policy

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's work or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result.

For more on the University of Pretoria's plagiarism policy, you may visit the following webpage: `http://www.library.up.ac.za/plagiarism/index.htm`

1.3 Practical component [25%]

You must first complete all 3 task of this practical. Once you have done so you must get marked by a tutor in the session. You will be asked to show what you have done for each task, and why. You must be ready to be marked 30 minutes before the end of the session.

1.3.1 Task 1: Input and Output [5 %]

For this task you must implement an assembler program that does the following:

- Display "Please input a 5 digit number: "
- Get a 5 digit input from the user
- Display "This is the number you are looking for: XXXXX" where XXXXX is the 5 digit number the user gave as an input.(Note: there should be a new line after displaying this sentence).

You will do these operations using system calls (syscall). The following link may aid you in understanding system calls:// `http://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/`//

Take note of which registers are used for system calls and which values the registers need to be set to in order to make the correct calls.

You should also possibly take a look at tasks from the previous Practical that used system calls in order to guide you in this Practical.

Example of the output should be as follows:

```
Please input a 5 digit number: 12345
This is the number you are looking for: 12345
```

You do not have to worry about the user entering more or less than 5 digits, or anything aside from numbers. We will have only non-idiot users test your code, this goes for the other tasks in this practical and assignment as well.

When you are finished, create a tarball containing your source code file named **task1.asm** and upload it to the assignments.cs.up.ac.za website, under the **Practical 2 Task 1** submission slot.

1.3.2 Task 2: Lower to Upper (GDB)[8 %]

For this task you must implement an assembler program that does the following:

- Display "Please input a lowercase character: "
- Get a single lowercase character.
- Display "In uppercase: X" where X is the now converted uppercase character of the lowercase character given to you by the user as an input.(Note: there should be a new line after displaying this sentence).

To see what the value of the character it might be useful to use GDB. GDB can help you see the value that is stored in a register at a particular point of the program and how you would need to change the value of the character in order for the character to change from lowercase to uppercase. Note: characters are stored as ASCII values, thus you would need to change the value in the register in order to change the character.

Refer to `https://www.asciitable.com/` for futher info on ASCII values.

Refer to `http://www.yolinux.com/TUTORIALS/GDB-Commands.html` for help with GDB.

Example of the output should be as follows:

```
Please input a lowercase character: h
In uppercase: H
```

When you are finished, create a tarball containing your source code file named **task2.asm** and upload it to the assignments.cs.up.ac.za website, under the **Practical 2 Task 2** submission slot. You must also demonstrate how you a achieved the result to a tutor for marks. How to use GDB will be asked and marks will be allocated towards using GDB.

1.3.3 Task 3: Add (GDB) [12%]

For this task you must implement an assembler program that does the following:

- Display "Please input the first number: "
- Get a positive integer value and assign it to a register.
- Display "Please input the second number: "
- Get a positive integer value and assign it to a different register.
- Add both values together and then display the sum of the input values. (This task will only test for the sum to be of a value that is less than 10)

Example of the output should be as follows:

```
Please input the first number: 2
Please input the second number: 3
5
```

Note that the values being entered would be stored as ASCII values and would need to be converted to decimal before adding them together. After you add the values together you will need to convert the sum back to an ASCII value in order to display it. Remember to make a newline after displaying the final sum.

When you are finished, create a tarball containing your source code file named **task3.asm** and upload it to the assignments.cs.up.ac.za website, under the **Practical 2 Task 3** submission slot. You must also demonstrate how you a achieved the result to a tutor for marks. How to use GDB will be asked and marks will be allocated towards using GDB.

1.4 Assignment component [75%]

For this assignment you will be required to implement 4 different single digit calculators. The calculators will only need to accept two numbers. Each calculator will handle one operation of either addition, subtraction, division or multiplication.

Your output should always be 2 digits. If your answer is a single digit pad your output with a zero. So for example 4+2=06.

You may assume that the numbers given to you will always be positive. You may also assume that no input given to you will result in a negative result.

1.4.1 Task 1: Addition [18.75%]

For this task you must extend Task 3 from the practical to handle double digit outputs.

Example:

```
Please input the first number: 8
Please input the second number: 6
14
```

When you are finished, create a tarball containing your source code file named **add.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 2 Task 1** submission slot.

1.4.2 Task 2: Subtraction [18.75%]

For this task you must take the 2 digits, subtract them and display the difference of the 2 numbers.

Example:

```
Please input the first number: 9
Please input the second number: 2
07
```

When you are finished, create a tarball containing your source code file named **sub.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 2 Task 2** submission slot.

1.4.3 Task 3: Multiplication [18.75%]

For this task you must take the 2 digits, multiply them together and display the product of the numbers.

Example:

```
Please input the first number: 3
Please input the second number: 9
27
```

When you are finished, create a tarball containing your source code file named **mul.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 2 Task 3** submission slot.

1.4.4 Task 4: Division [18.75%]

For this task you must take the 2 digits, divide them and display the result with the remainder in the following way.

Example:

```
Please input the first number: 7
Please input the second number: 2
3r1
```

When you are finished, create a tarball containing your source code file named **div.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 2 Task 4** submission slot.

2 Mark Distribution

Activity	**Mark**
Prac Task 1	5
Prac Task 2	8
Prac Task 3	12
Assignment Task 1	18.75
Assignment Task 2	18.75
Assignment Task 3	18.75
Assignment Task 4	18.75
Total	**100**