CISC 458
Tutorial 1

Project Overview, Getting Started, and Examples

Revised by Ahmed Harby from Previous Versions

Thursday, January 19, 2023

Overview

TAs Information

- The TAs for this course are:
  - **Ahmed Harby**
    - Ahmed.Harby@queensu.ca
  - **Harshkumar Patel**
    - patel.h@queensu.ca
  - **Chunli Yu**
    - chunli.yu@queensu.ca
  - **Mohamed Ads**
    - m.ads@queensu.ca
- The TAs are responsible for:
  - Project tutorials
  - Help sessions/advising
  - Marking projects

Contact Time & Locations

- Tutorial Schedule
  - The tutorials will be held in Stirling Hall B on Thursdays 18:30-20:30 starting from Jan 19.
- Project Advising Schedule
  - The majority of tutorials will not exceed an hour.In the remaining time, I will be available for project advice.
  - There will be three additional advising sessions assigned by each TA (details to be shared soon).
- Electronic Help Through OnQ & Email
  - Any time (reply time may vary)
  - **OnQ discussion board preferred**

Computing Resources

- We will be using the CASLab Linux Servers
    - `linux[1,2,3,4,5,6].caslab.queensu.ca`
- The servers can also be accessed using any SSH client (e.g., ssh, PuTTY). You can find all the details on how to setup SSH client on "https://courses.caslab.queensu.ca/"
- Dr. Karim will set up group accounts for each group.
    - `Group names will be of the form cisc458_x`, where `x` is an integer .
- Unfortunately, that's the only way to work on your project
    - Edit files on your machine and push them to the server using SCP or SFTP (e.g., scp, WinSCP, FileZilla).
    - You can also use version control tools like Git.

Getting Started

Getting the PT Compiler

You need to copy the compiler's source to your home directory

```
cd ~
mkdir cisc458
cd cisc458
tar xvf /cas/course/cisc458/pt23/pt23-student.tar.gz
```

Now you have a copy of the PT Compiler that you can modify all you want!!

Building the Compiler

Once you have the source unpacked, you can build the compiler:

```
cd ptsrc
make install
```

- `make` essentially builds the compiler for you! It will take care of knowing what has changed, and recompile and copy the files.
- You will need to remake the project whenever you make any changes to the source.
  - There is also a make file in every sub folder.

The PT Library

The `make` command will copy the necessary files into your new library.
The files will be copied to:

```
ptsrc/lib/pt
```

This is where **your** version of the library is kept.

Recap

- You should now have the following directories:
    - `~/cisc458/ptsrc` contains all the source code for the compiler. This is where you will make changes.
    - `~/cisc458/ptsrc/lib/pt` is where the library built using the `make` command is stored. You will need to point to this library when running the compiler.
- Remember to run `make` whenever any changes are made to the source code!

Running the Compiler

- When running the compiler, you must point it to your custom library.
- If you do not specify a library, the default one will be used.
- You can run the compiler with the following commands:

```
# uses the default library
ptc <source file>
# uses your custom library
ptc -L ~/cisc458/ptsrc/lib/pt <source file>
```

More about Libraries

- The -L flag specifies the library directory, so you can create more than one directory if you want.
- Annoying mistake is forgetting the -L flag and running the default library!
- You might want to create an alias or a simple script for it.

```
# aliasing of custom library
alias ptccl='ptc -L ~/cisc458/ptsrc/lib/pt'
```

- You can add the alias command to ~/.bash_profile to make it permanent.
- Alternative Options:
    - ~/ptsrc/test contains a shorthand command ./ptc which uses the custom library.

PT Pascal Examples

What is `PT`?

- `PT Pascal` is a subset of `Pascal`.
- A full definition of the language is in Appendix A1 of the last section ("PT: A Pascal Subset") of your textbook.
- Major syntactic differences from Java/C:
    - Variables are declared as **name : type**
    - **:=** is the assignment operator
    - = is reserved for equality
    - Strings uses single-quotes **('...')** instead of double-quotes **("...")**.

Hello World Example

```pascal
program HelloWorld ( output );

{ This is a very simple program which
    just prints the string "Hello World" }

begin
    write('Hello World');
    writeln
end.
```

Variables Example

```
program VariablesExample ( output );
var x : integer;
begin
    x := 10;
    write('The value of x is ', x);
    writeln;
end.
```

Input Example

```
program InputExample ( input , output );
var x : integer;
begin
    write('Enter a number: ');
    read(x);
    write('The number was ', x);
    writeln;
end.
```

If Example

```
program IfExample ( input , output );
var x : integer;
begin
    write('Enter a number: ');
    read(x);
    if x > 0 then begin
        write('The number is greater than zero');
        writeln
    end
end.
```

Course Project

First Steps

- Find a team of 4 members and sign the Team Agreement.
- Familiarize yourself with PT Pascal
    - Get & build the source code.
    - Compile & run the examples in this tutorial.
    - There are more examples in `ptsrc/examples`.
- Understand the **Quby** language

Next Tutorial

- In the next tutorial, we will discuss the modifications to the compiler for **Phase 1**.
    - You can start looking at the code for the scanner (`ptsrc/parser/scan.ssl`).
- During the tutorials, we will give out a lot hints for each phase.
- Don't get stuck! If you find yourself struggling with certain changes, please post on OnQ!
    - Please avoid posting one hour before the deadline!
    - Please remember to ask **all** your teammates for input!

Demo