

Understanding Ctypes

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 4](#_Toc65424206)

[Background Reading 4](#_Toc65424207)

[Important Information 4](#_Toc65424208)

[Introduction 5](#_Toc65424209)

[You will need Windows to try the next few lines 7](#_Toc65424210)

[Questions (answer in your own words. Do not copy+paste the python documentation) 7](#_Toc65424211)

[You will need Linux to try the next few lines: 8](#_Toc65424212)

[Questions 8](#_Toc65424213)

[Guru Points (Note this will not provide any extra points) 9](#_Toc65424214)

[CTYPES by any other name … “A pain in my … head” 9](#_Toc65424215)

[Scenario: 9](#_Toc65424216)

[Questions 11](#_Toc65424217)

[Building a library 12](#_Toc65424218)

[Compile the program and create a shared library named libprintproc.so. 13](#_Toc65424219)

[Questions: 13](#_Toc65424220)

Offensive and Defensive Tool Construction

Debugging Theory

Objectives

This lab focuses on the following objectives:

* Explain ctypes
* Explain C data structures (structure, union and arrays) as they relate to ctypes
* Explain and show how to use ctypes
* Implement the concepts of ctypes

Background Reading

Read chapters 1 in the *Gray Hat Python* textbook. The following links are also useful:

* https://docs.python.org/3/tutorial/interpreter.html
* https://en.wikipedia.org/wiki/Microsoft\_Windows\_library\_files
* https://docs.python.org/3/library/ctypes.html
* https://docs.python.org/3.8/extending/extending.html

# Important Information

**YOU MUST PRESENT IMAGES OF YOUR CODE BEING EXECUTED. DO NOT SUBMIT YOUR ANSWERS IN THE DOCUMENT. CREATE A BLANK DOCUMENT AND SUBMIT YOUR ANSWERS THERE.**

**YOU WILL LOSE MARKS FOR NOT FOLLOWING THE ABOVE REQUIREMENTS.**

All scripts must have the following elements:

1. File and Header comments, which follows the following format:

***# Filename: m##XXX.py***

***# Author: Thunder Cat***

***# Course: ITSC203***

***# Details: This exercise checks to see if students read the suggested items or***

***# prior to class or doing the labs.***

***# Resources: https://www.cs.siue.edu/programming-style-guide***

1. Comments on lines where you used some unique computation that might be tricky to comprehend a month later.

***list1 = [x for x in range(20) if x % 4 == 1] # Using list comprehension to ….***

# Introduction

In applications that make requests to low level facilities, there are times when writing Python code may not get the job done. More precisely, it may not get the job done as quickly as natively compiled code.

There are also times when prebuilt, task specific libraries are already available on the system. In those cases, rolling our own python implementation would be redundant and potentially buggy. Instead, we should use the available libraries, written by SME (subject matter experts) to save ourselves some time. Thus allowing us to stand on the shoulders of giants in the industry.

How can we access printf or scanf, or any interesting low-level Operating System APIs we might want to access. The answer, we need a way to cross over from Python to C implemented libraries. There are modules available for this type of task that are collectively referred to as Foreign Function Interfaces (FFI). One such module built-in and readily available is the python ctypes module.

Before delving into the ctypes module lets first revisit and reintroduce a few things from C. Last semester you were introduced to structures. The structure is a powerful abstraction of many types of data components within the Operating System. For example, it is used to hold data transferred between functions. It is also used to relay information about processes and is also used for creating “Linked Lists”; which is to track and maintain a list of system processes.

Here is an example of a structure:

**typedef struct  
{  
 int ppid;  
 int pid;  
 char name[10];  
 char children[10][30];  
}process;**

The structure above is a contrived (abstract) representation of a process. We can see here that this struct groups together different datatypes for the purpose of keeping track of some information (name, process ID etc) related to a process.

C also has a data structure known as the UNION. If this is the first time you have seen this, the easiest explanation for it is as follows. Let’s say you received data from the user but sometimes that data could be a letter, a number type (float or integer). The union allows you to assign a value to **ONLY one member** of the UNION at any point in time. So, in the example described, the program can assume the identity of either an int, float or char.

The format of the union is similar to the struct as show below:

**typedef union  
{  
 int value1;  
 char value2;  
 char name[10];  
}process;**

To see the structure and union in action type in the following code and investigate the behavior of the code. Switch the order of lines 20 and 21 around and observe the output. Based on the description of the union provided answer the following question.

## Question (no marks)

1. Why does changing the order of lines 20 and 21 produce different output? *See the output of line 23 on the terminal.*



Although not explored in this document, remember there are data structures in C; think of the array. There are also many variable types useful for holding data, example int, float and char.

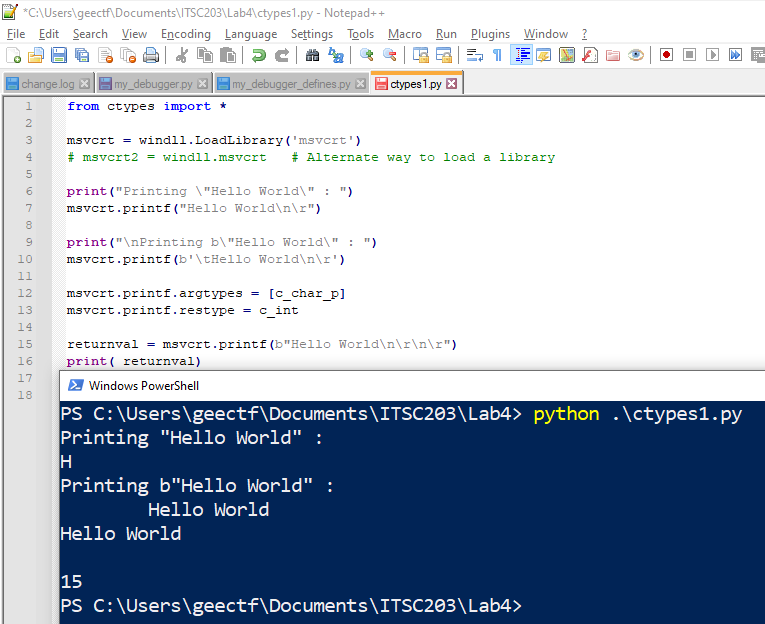
Now that we understand the concept of these data structures, let’s explore some ctypes!!!

# You will need Windows to complete the next few lines

*Submit screen shots of the work that you did to accomplish each task.*

1. Install python3.8+ on your Windows VM
2. Install a Text Editor. Options include (Notepad++, sublime, Visualstudio Code)
3. Install PyCharm
4. Open a text editor and enter the code below:
5. Execute the program from the Powershell window below: **python .\ctypes1.py**
6. Alternately you can use the IDE you installed earlier.

*Note: There are only 16 lines of code to copy. The Powershell window is not hiding any code.*



## Question 1 (5 pts)

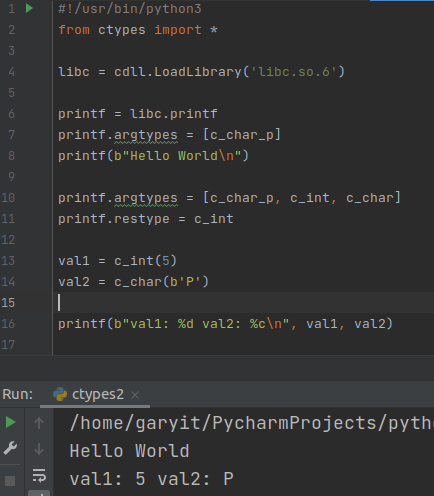
**(answer in your own words. Do not copy+paste the python documentation)**

1. Why does **Line 6** produce the output shown in the Powershell window above? (**1pt**)
2. What does the **'b'** in front of the string object on line 10 do? (**1pt**)
3. What is the purpose of **argtypes** and **restype**? (**1pt**)
4. The last line of the output shows the number **15**. What is the significance of this number? In other words where did it come from? (**1pt**)
5. Write the command you would use to load the **kernel32.dll** a ***Dynamically Linked Library***. (**1pt**)

# You will need Linux to try the next few lines:

*Submit screen shots of the work that you did to accomplish each task.*

1. Open a text editor, I used PyCharm seen below. Enter the following code:
2. Execute the command from Python as shown or from the command-line with the command: **python3 .\ctypes1.py**



## Question 2 (5pts)

1. What is **libc.so.6** on line 4? (**1pt**)
2. What is the absolute path of the location of the file **libc.so.6**? (**1pt**)
3. Which variables in the function **printf** are **c\_char\_p**, **c\_int** and **c\_char** linked to? (**1pt**)
4. How would you modify the code so that you can print a third value, val3? (**1pt**)
   1. val3 is a string: "You got it …"
   2. val3 must be a separate string from the format string.

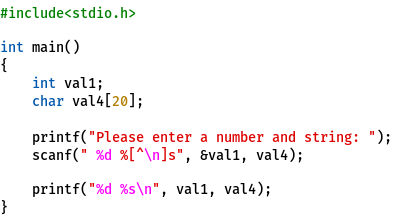
**The following is not a valid solution:** **printf(b'val1: %d val2: %c val3: You got it…\n", val1, val2)**

1. Provide the modified Line 10 in your write-up that completes the solution for Step 4. (**1pt**)

## Guru Points (Note this will not provide any extra points)

This is a bit challenging, as the solution takes a bit of time to work out. In other words, if you don’t have time don’t dedicate too many cycles trying to find a solution.

1. Convert the following code to python using the ctypes module:



# CTYPES by any other name … “A pain in my …”

In the other sections we saw that ctypes can access a library of our choosing and we can transmit some data to that library and get some response back. I suspect that the idea is still a bit abstract.

So, let’s try to understand the entire process by building the following components:

1. A simple shared library, which provides our program a function
2. Some python code to make use of that library

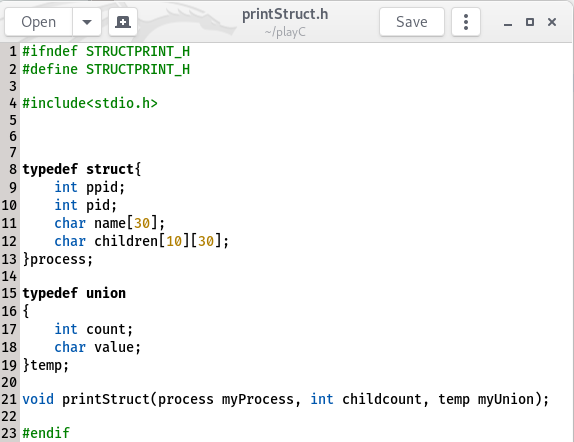
***Submit screen shots of the work that you did to accomplish each task.***

## Scenario:

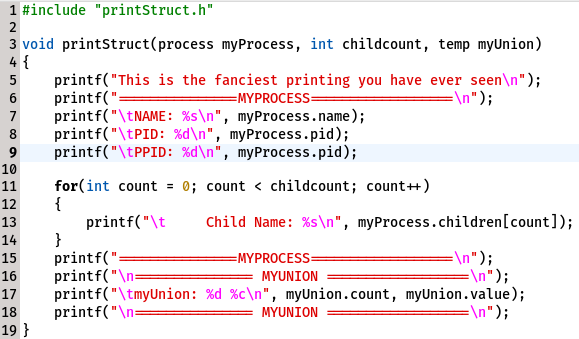
Let’s us say that there is a library called printStruct. It’s main purpose in life is to print the data inside a structure and union. The output will be formatted using a fancy table to present the information. Imagine that only our C library is capable of this feat!!!

To accomplish this task let’s do the following:

1. Create the **printStruct.h** header file. All C files in our project will use this file for constants and prototypes.

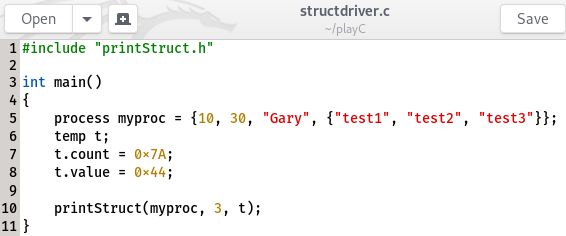


1. Create the **printStruct.c** file. This file contains a single function to print data in the struct.

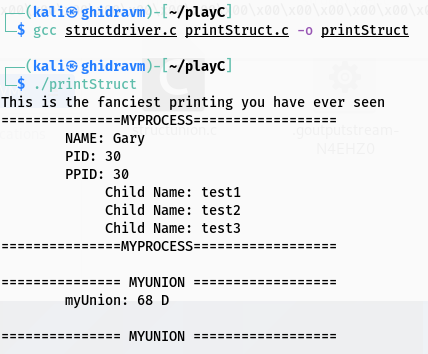


1. Create the **structdriver.c** file to test the above code let’s create some driver code that will call printStruct, provide some information that will be printed.

This is just test code that we will attempt to implement in python later to test our library.



1. Compile the code and execute the program. See the steps in the image below:



## Question 3 (8pts)

***Submit screen shots of the work that you did to accomplish each task.***

1. What is the purpose of the printStruct.h file? (**2pts)**
2. What is the purpose of the printStruct.c file? (**2pts)**
   1. Does the file need a main function?
3. What is the purpose of the structdriver.c file? (**2pts)**
4. As shown, the compiler can process multiple C files together to create a single (**2pts)** executable.
   1. Is it possible to create a separate C file, that can be used to print ONLY the data in the union?
   2. Write the code that would be contained in the file to print only the Union. You can call the file printUnion.c?
   3. Take screen shots to show the process of compiling, and executing the 3 .c files.

## Building A Library

Follow the steps below to create the library **libprintstruct.so**

When we execute our C programs we are making use of the **libc** functions available to us. **Libc** is a shared object (SO) file. Shared objects means that our program will share the library with other programs that may also need that program.

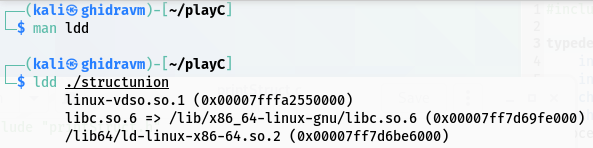
For example, if there are 2 processes running on your computer that need to print, they will both make requests of the print function located in memory. The printf function exists as part of the libc ecosystem and is provided for general use.

When the processes are loaded the OS through the use of its loader checks all dependencies including any libraries required for execution. In our case, **libc.so.6** (aka libc) is one such library that is needed.

**NOTE:** The alternative to **Shared Object** libraries is static libraries.

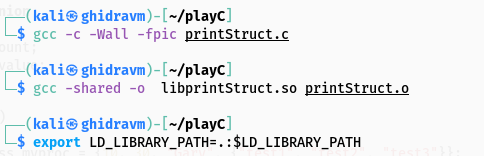
Can you gather from the name why they are called static?

See the dependencies that are needed for your executable:

**WARNING**: ldd executes the program. Therefore, if the ELF is evil you could execute some arbitrary code; this could lead to a bad thing happening.

## Compile the program and create a shared library named libprintStruct.so.

1. gcc -c -Wall -fpic printStruct.c
2. gcc -shared -o libprintStruct.so printStruct.o
3. export LD\_LIBRARY\_PATH=<path of libprintStruct.so>:$LD\_LIBRARY\_PATH



## Question 4 (9pts)

***Submit screen shots of the work that you did to accomplish each task.***

1. What do the options -shared, -c, -Wall, -fpic mean? (**3pts)**
2. Where is the file **libprintStruct.so** located? (**3pts)**
3. What does the purpose of the command (**3pts)**

**export LD\_LIBRARY\_PATH=.:$LD\_LIBRARY\_PATH?**

## Build the Python test library program

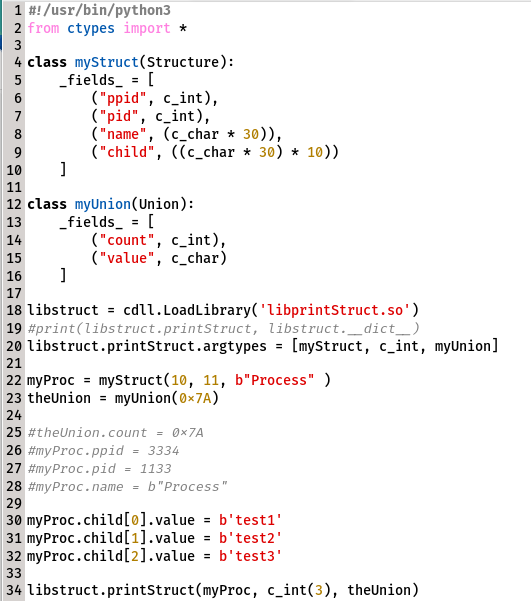
***Submit screen shots of the work that you did to accomplish each task.***

Now that we have the library, it is time to test our ability to load this library and perform calls to it using CTypes.

Copy the following code and test it:

**IMPORTANT**: **If you attempt to use PyCharm to test this code, you NEED to setup an environment variable called LD\_LIBRARY\_PATH. Otherwise, you will have to use the absolute path of the library file on line 18 in the code.**

1. Create the printStruct.py file shown below; and execute the python program



## Question 5 (10 pts)

***Submit screen shots of the work that you did to accomplish each task.***

1. Lines 22 and 23 are each performing an initialization. What variables are being initialized with the values **10, 11, b'Process' and 0x7A**? (**2pts)**
2. Enumerate (or List) the steps required to add a one-character variable, that represents state, in the struct myStruct. The state can be either Z, I, R and S for (**Zombie, Idle, Running and Sleeping**) respectively. (**2pts)**
3. Add a function called printState to the **libprintStruct.so** library (**2pts)**
   1. The **printState** function will be called from the **printStruct** function and takes only one argument (a character representing the state).
   2. The function will then print the message about the status as a word instead of the letter that was provided.
4. Can the **printState** function be called directly from the python driver? (**2pts)**

Show an example statement that could be used to perform a direct call to the **printState** function.

1. What is the advantage of using the python driver above, instead of the C driver created earlier? (**2pts)**

**Hint:** think about if you had to modify parameters.