CSC344 Problem Set: Memory Management / Perspectives on Rust

Bryan McLean

Task 1: The Runtime Stack and the Heap

In Rust programming, the stack and the heap are both parts of the memory that is available to the program during runtime. Data that is stored in the stack has a known size that is fixed when the program is compiled. Data that is stored in the heap has an unknown size when the program is compiled or if the size of the data might change then the data must be stored in the heap. Rust uses the runtime stack and the heap instead of other types of memory control because other types can either make the program execute slower or give the user little memory management control.

When a rust program is run, a stack is created which stores everything happening in the program. The main method gets added to the stack as well as every function that is run with its variables with a defined size of data. When a function is finished being executed, its variables and values for the variables get thrown out of the stack.

Any large collection of data is stored in the heap in rust and has a reference to the data in the stack. The heap is slower than the stack which is why most data is stored in the stack. The heap is unlimited in size and can be accessed globally unlike the stack.

Task 2: Explicit Memory Allocation/Deallocation vs Garbage Collection

Many other programming languages use explicit memory allocation which means that the programmer must create and free memory manually while programming in a language that uses explicit memory allocation. Garbage collection is a form of automatic memory management that stores allocated memory that is no longer being used by the program. Both methods have advantages and disadvantages to them.

C++ is an example of an explicit memory management language. In these types of languages, the user must manually allocate and deallocate memory for variables and functions. When a programmer allocates memory, they need to make sure that they deallocate all that memory at the end of the program because if they do not then memory leak can occur.

Haskell is an example of a language that uses a garbage collector. A garbage collector stores allocated memory that is no longer being used by the program. Using one can help to avoid issues such as memory leak and dangling pointers. The garbage collector can make a program take longer to run which can make a big difference if a program needs to be executed very quickly.

Task 3: Rust- Memory Management

“The suggestion was that Rust allows more control over memory usage, like C++.”

“In Rust, we do allocate memory and de-allocate memory at specific points in our program. Thus it doesn't have garbage collection, as Haskell does. But it doesn't work quite the same way as C++.”

“Heap memory always has one owner, and once that owner goes out of scope, the memory gets de-allocated.”

“Rust works the same way. When we declare a variable within a block, we cannot access it after the block ends.”

“What's cool is that once our string does go out of scope, Rust handles cleaning up the heap memory for it!”

“Memory can only have one owner.”

“Passing variables to a function gives up ownership.”

“Like in C++, we can pass a variable by reference. We use the ampersand operator (&) for this. It allows another function to "borrow" ownership, rather than "taking" ownership.”

“You can only have a single mutable reference to a variable at a time”

“Slices are either primitive data, stored on the stack, or they refer to another object. This means they do not have ownership and thus do not de-allocate memory when they go out of scope.”

Task 4: Paper Review- Secure PL Adoption and Rust

Many memory safety vulnerabilities can occur in some programming languages such as C and C++. These vulnerabilities can arise from violations of memory safety. C and C++ both use explicit memory management which leads to more vulnerabilities because the user has to allocate and deallocate memory manually for the data in a program. To help to resolve this problem programming languages such as Go and Rust were recently developed.

Rust does not use garbage collection like many other programming languages. Rust is designed to be fast and much safer for memory than C and C++. Since Rust has been developed, it has become very popular with clear error messages and very good documentation. Rust has many important features such as it has traits that can be used to encode objects, variables are immutable by default, and cargo which downloads necessary library packages for your code.

Although Rust has many positives, it also has many negatives such as the language having a steep learning curve to adjust to the security, limited library support and compiling taking a long time. Since it is a new language, a lot of these issues may be fixed in the future and will improve the language. Some companies are also reluctant to adopt Rust because they don’t know the future of the language and are unsure if it would be beneficial to switch to Rust from the languages that they are currently using.