SNS COLLEGE OF TECHNOLOGY

COIMBATORE –35

(An Autonomous Institution)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# Run-Time Environment & Source Language Issues

A program as a source code is merely a collection of text (code, statements etc.) and to make it alive, it requires actions to be performed on the target machine. A program needs memory resources to execute instructions. A program contains names for procedures, identifiers etc., that require mapping with the actual memory location at runtime.

By runtime, we mean a program in execution. Runtime environment is a state of the target machine, which may include software libraries, environment variables, etc., to provide services to the processes running in the system.

Runtime support system is a package, mostly generated with the executable program itself and facilitates the process communication between the process and the runtime environment. It takes care of memory allocation and de-allocation while the program is being executed.

## Activation Trees

A program is a sequence of instructions combined into a number of procedures. Instructions in a procedure are executed sequentially. A procedure has a start and an end delimiter and everything inside it is called the body of the procedure. The procedure identifier and the sequence of finite instructions inside it make up the body of the procedure.

The execution of a procedure is called its activation. An activation record contains all the necessary information required to call a procedure. An activation record may contain the following units (depending upon the source language used).

| | |
|---|---|
| Temporaries | Stores temporary and intermediate values of an expression. |
| Local Data | Stores local data of the called procedure. |
| Machine Status | Stores machine status such as Registers, Program Counter etc., before the procedure is called. |
| Control Link | Stores the address of activation record of the caller procedure. |
| Access Link | Stores the information of data which is outside the local scope. |

| | |
|---|---|
| Actual Parameters | Stores actual parameters, i.e., parameters which are used to send input to the called procedure. |
| Return Value | Stores return values. |

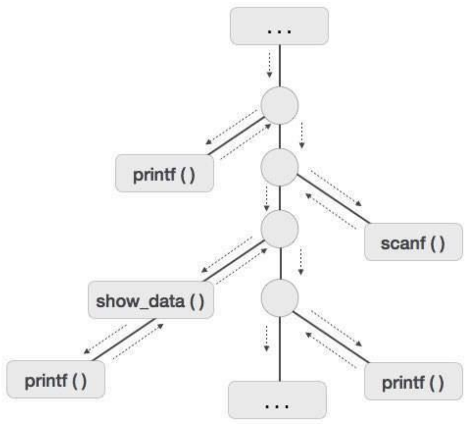
Whenever a procedure is executed, its activation record is stored on the stack, also known as control stack. When a procedure calls another procedure, the execution of the caller is suspended until the called procedure finishes execution. At this time, the activation record of the called procedure is stored on the stack.

We assume that the program control flows in a sequential manner and when a procedure is called, its control is transferred to the called procedure. When a called procedure is executed, it returns the control back to the caller. This type of control flow makes it easier to represent a series of activations in the form of a tree, known as the **activation tree**.

To understand this concept, we take a piece of code as an example:

```
. . .
printf("Enter Your Name: ");
scanf("%s", username);
show_data(username);
printf("Press any key to continue…");
. . .
int show_data(char *user)
  {
  printf("Your name is %s", username);
  return 0;
  }
. . .
```

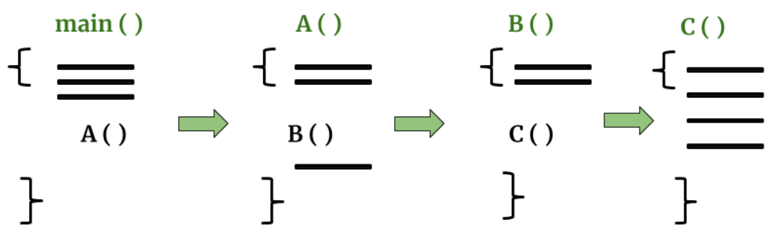
Below is the activation tree of the code given.



Now we understand that procedures are executed in depth-first manner, thus stack allocation is the best suitable form of storage for procedure activations.

## CONTROL STACK AND ACTIVATION RECORDS

Control stack or runtime stack is used to keep track of the live procedure activations i.e the procedures whose execution have not been completed. A procedure name is pushed on to the stack when it is called (activation begins) and it is popped when it returns (activation ends). Information needed by a single execution of a procedure is managed using an activation record or frame. When a procedure is called, an activation record is pushed into the stack and as soon as the control returns to the caller function the activation record is popped.



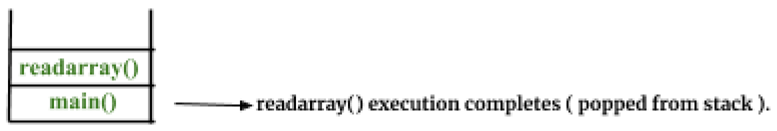
Before going to A() activation record of main is created when A() is called activation record of A() is created.

Before going to B() activation record of main and A() in stack.

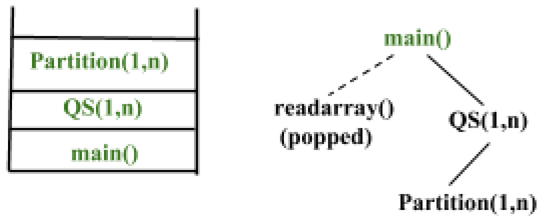
Before going to C() activation record of main and B() is pushed in stack.

A general activation record consist of the following things:

- Local variables: hold the data that is local to the execution of the procedure.
- Temporary values: stores the values that arise in the evaluation of an expression.
- Machine status: holds the information about the status of the machine just before the function call.
- Access link (optional): refers to non-local data held in other activation records.
- Control link (optional): points to activation record of caller.
- Return value: used by the called procedure to return a value to calling procedure
- Actual parameters

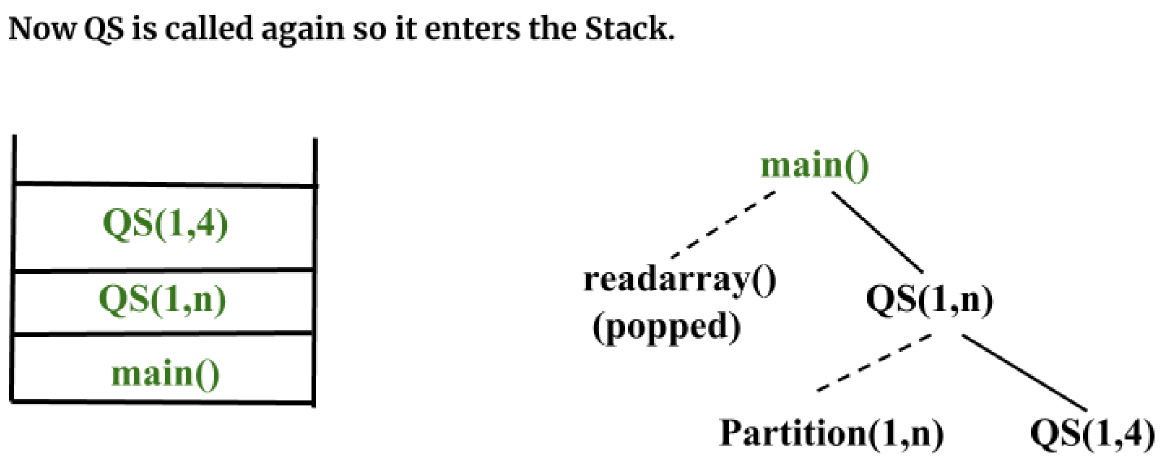


QS is called so it Enters the Stack.



Partition Execution completed (popped out of stack)

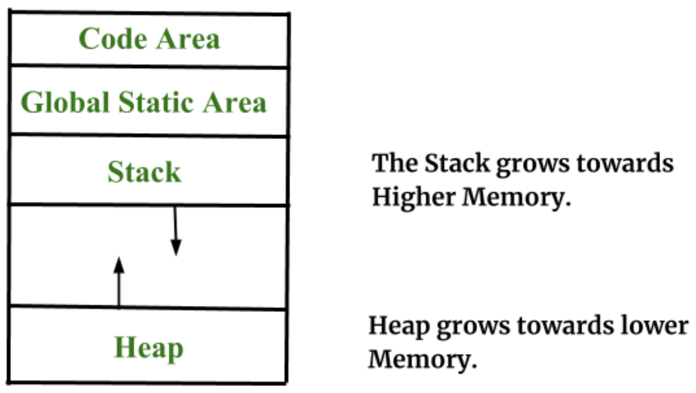
Control stack for the above quicksort example:



SUBDIVISION OF RUNTIME MEMORY

Runtime storage can be subdivided to hold :

- Target code- the program code, it is static as its size can be determined at compile time
- Static data objects
- Dynamic data objects- heap
- Automatic data objects- stack


Code Area
Global Static Area
Stack
Heap
The Stack grows towards Higher Memory.
Heap grows towards lower Memory.