Minor Project- Report

Apr 2021-Jul 2021

Course Faculty: Prof. Prameetha Pai Course Name: System Software

Course code: 18CS6DCSSW

Semester: 6 Date:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TITLE OF THE PROJECT | LINKER IMPLEMENTATION | | | |
|  |  | | | |
| STUDENT NAME | JENNIFER S JEEYAR | K R ROHIT SRIVASTAVA | KANISHKA SHAH | KARTHIK WADEYAR |
| USN | 1DS18CS053 | 1DS18CS054 | 1DS18CS055 | 1DS18S056 |
| INDIVIDUAL  CONTRIBUTION | Preparing algorithm and base structure | Implementation of algorithm in C language | Programming in C and testing | Preparing I/O test samples and debugging |
| GUIDE | Prof. Akshata | | | |
|  |  | | | |
| PROJECT ABSTRACT : | In computing, a linker also called as link editor is a computer system program that takes one or more object files which is generated by the compiler or assembler and combines them into a single executable file. The linker also takes care of arranging the objects in a program’s address space.  The general aim of this project is to simulate the working of a linker. As it is an important component in the stages of execution, its working and fundamental concepts will be highlighted and implemented through this project.  The project also demonstrates the implementation of the two main types of linking, ie,   1. Dynamic linking 2. Static linking | | | |
| PLATFORM USED  (H/W & S/w tools to be used | C language, Visual Studio Code | | | |
|  |  | | | |
| Introduction | In computer systems linker is a computer system program that takes one or more object files and combines them into a single executable file.  Linking is a process of collecting and maintaining a piece of code and data into a single file. Linker also link particular module into system library. It takes object modules from assembler as input and forms an executable file as output for loader.  Linking is performed at both compile time, when source code is translated to machine code and load time, when the program is loaded into memory by loader.    ***Source Code->Compiler->Assembler->Object code->Linker->Executable File->Loader*** | | | |
|  |  | | | |
| Design | ***Data Flow Diagram***    H, R, D, T, M, E  Records Intermediate T, M Records  Storage  Source code (From assembler)  Object Pass 1 Pass 2  code Modified  object code  (Symbol, Address)  Load Map  ESTAB (Symbol, Address)    In first pass External Symbol Table(ESTAB) is generated. This can also be accessible later by pass 2. It is used to store the name and address of each External Symbols in a set of Control Sections and also it assigns address to all External Symbols. It has two variables PROGADDR and CSADDR. PROGADDR is the beginning address in memory where the linked program is to be loaded, and CSADDR contains the starting address assigned to the control section currently being scanned by the loader.  Pass 2 linking loader performs actual loading, relocation, and linking. It uses modification record and lookup the symbol in ESTAB to obtain its address. Finally it uses end record of a main program to obtain transfer address, which is a starting address needed for the execution of the program. The pass 2 process Text record and Modification record of the object programs. | | | |
|  |  | | | |
| Project Source Code Link (Github/ Google DRive) |  | | | |
|  |  | | | |
| Conclusion /FUTURE ENHANCEMENT |  | | | |
|  |  | | | |
| Ui sCreenshots | Text  Description automatically generated  Fig 1: Input consisting of 3 object Programs.  Text  Description automatically generated  Fig 2: Output given by pass-1  Graphical user interface  Description automatically generated with medium confidence  Fig 3: Output given by pass-2 | | | |