OS PROJECT REPORT
Team-3

Project Introduction and Its Features :

Main Aim of our project was to demonstrate the minimizing effects on different processes. Users can open Different Applications in our modified Xinu OS and can have a feel of minimizing processes and opening them again. Different Applications which will be available are Reminder , Calculator , Creating/editing files , Todo List . So using specific keyword like **'#'** users can minimize any process at any point of time. And Users can have a look at different minimized processes by using same keyword **'#'** in the main process and will be able to open them again seamlessly.

Feature not implemented : -
Feature based on Shared memory concept where Users can create two text files and link them and those two files will share the same data.

Team Members Contribution :

Hemant Kumar (CS18B014) - process scheduling, process minimization, application like - reminder, calculator, file editor (partial).

Rohit Shakya (CS18B029) - memory management, application like - file editor, calculator (partial).

Sourabh Misal (CS18B022) - applications like - todo-list and file editor (partial).

K Pavan Kumar (CS18B015) - applications like - todo-list (partial) and file editor (partial).

Final Top - Down details of project

So, In the beginning we pushed ourselves to complete applications like calculator, reminder, creating/reading files, Todo-List, sharing a file (both files will reflect each other). But We have successfully completed only applications like calculator, reminder, creating/reading files, Todo-List. So we are not able to do file sharing applications.

Final Applications which can be run on our modified Xinu OS

A Simple Calculator
A simple calculator for calculations.

Reminder
Reminder with task description and time, which will prompt after time goes down to zero.

File-editor
Simple file-editor, where we can create new files and here we will be having different other functionalities like update/read/delete . We can minimize file editor while updating content of a file, and can reopen it for saving unsaved content.

Todo-List
Any number of tasks could be attached to the Todo-List.

Minimization
Minimization feature is implemented which can minimize applications described above.

Final Bottom - Up details of project

- **OS feature: Process Scheduling**

So for Scheduling/Rescheduling different minimized processes we have used process scheduling in our project to schedule different processes according to users.

- **OS feature: Context Switch**

When our OS will transit from one process to another

- **OS feature: Memory Management**

When we will do context switching, we need to store our previous processes data somewhere, for which we will use memory management. Processes like

creation of files or deletion of files all require memory management.
Memory management is also needed for storing history of calculations, storing data of an application that needs to be saved.

● **OS feature: Process Management**
For managing different processes we have used process management to carefully run different processes without getting into traps.

Implementation of different features

S.No.	Project Feature description	Xinu files modified	Xinu files used	New files added
1.	**Calculator** : Calculator is a process which can be minimized and opened again or a new calculator could also be opened instead of opening a minimized calculator.we can also see history of calculation.	Main.c , bufpool.h ,	Getbuf.c, freebuf.c, getpid.c , insert.c , resched.c , kprintf.c , Dequeue.c , Ready.c , Resume.c , Ctype.h , String.h , queue.h	reschedule.c ,
2.	**Reminder** : Reminder is a process where you can set a reminder with a description of it and it will prompt after given time. This process can also be minimized and open again.	Main.c ,	getpid.c , insert.c , resched.c , kprintf.c , Ready.c , Resume.c , Ctype.h, queue.h	reschedule.c ,
3.	**Todo-List** : Todo-list is an application where you can list all of your tasks which you want to complete and while writing all this you can minimize it and	Main.c	getpid.c , insert.c , resched.c , kprintf.c , Ready.c , Resume.c ,	reschedule.c ,

	open it again.		Ctype.h , queue.h	
4.	**File-editor :** File editor is an application where we can create new files and here we will be having different other functionalities like update/read/delete . We can minimize file editor while updating content of a file, and can reopen it for saving unsaved content.	Main.c, bufpool.h	getbuf.c, Freebuf.c, Strncat.c, strcpy.c, getpid.c, insert.c , resched.c , kprintf.c , Ready.c , Resume.c , Ctype.h	
5.	**Minimization** : This feature handles minimizing different processes and scheduling them when the user tries to open them again.	Main.c,	getpid.c , insert.c , resched.c , kprintf.c , Ready.c , Resume.c , Ctype.h , queue.h	reschedule.c ,
6.	**File Sharing (NOT IMPLEMENTED)** : We wanted to implement file sharing where users can create two files and link them so that they can hold the same memory. So if user writes something inside one while then other will reflect the changes			

Minimization Feature Implementation :
When a user minimizes some application then we are just storing those process id's of all those processes which got minimized and we are not inserting this process inside the ready list. And after minimizing we are invoking our main screen process using a new reschedule function which is for scheduling main screen process after minimizing any application which will again show options to open new applications. So when a user again tries to open a process which is inside a minimize list at that time we are just inserting that particular process inside the readylist which the user wants to open and calling normal process scheduler to open the process from the readylist.

Reminder Feature Implementation :

So when a user sets a reminder then we are storing that time inside a variable and then whenever a rescheduling function is called we are subtracting the time slice of that old process (which is QUANTUM - preempt) from the variable storing time for reminder to prompt. And when that time goes down to zero at that point of time we are prompting reminder.

File-editor Feature Implementation :
First we modified bufpool.h file, so that we can use high level memory management in xinu. Then We created a linked list to store data involved in different files in bufferpool. Then We ask users to perform operations like - create/update and change data of bufpool accordingly.

File Sharing Feature (NOT IMPLEMENTED) :
We wanted to implement this process by having a memory which can be used by two different files so both these files will point to the same memory and when we will write in one file then other will also show it as the memory is the same for both files.