

## 4. FEASIBILITY STUDY

### 4.1 *Objective:*

To design and implement an operating system that enables Internet access by providing the world-wide web services on low-end machines..

### 4.2 *Focus:*

The project aims to focus at the following main issues:

1. Enabling the web
2. Workability on low-end target systems
3. Light-weight, in not demanding high hardware/software requirements

### 4.3 *Project Constraints:*

The following constraints are inevitably imposed on the project: -

1. Lack of financial investment  
Being a project primarily of academic interest, the project does not involve monetary investment, and may not find financial aid.
2. Limited project time  
The project is a term-work requirement. The total allotted time for the project design and implementation is 15 weeks.
3. Target hardware  
The project is deliberately targeted at low-end systems with the following minimal hardware requirements (standard IBM-PC or clones with):
  - Processor: 486DX or Pentium (33MHz)
  - Primary Memory: 16 MB
  - Hard-disk storage: 200 MB or lesser
  - Networking capability: Network-interface card or Modem
  - Basic I/O devices: Keyboard and VDU
  - Graphics: VGA – capable

### 4.4 *Technical Issues:*

The following technical issues (and options available for them) must be considered for the project: -

- Whether or not such a system is possible
- The operating system to be chosen as the model for development
- What all software to develop.

1) Whether or not such a system is possible: -

Contemporary operating systems like MS-Windows 98 and Linux can be used on the present-day PC-configurations. The bare minimum requirements for them, in terms of hardware, are as follows: -

Hardware	MS-Windows 95 <sup>1</sup>	MS-Windows 98	Linux
RAM	8 MB	16 MB	2 MB
HDD space	40 MB	60 MB	20 MB
Processor	80386	80486DX	80386
Graphics	SVGA	SVGA	CGA/VGA

The above stated requirements clearly suggest that these operating systems can be scaled down to work on low-end machines, as required by the target-hardware constraints.

If a new operating system is written from scratch that does not implement any other user-service than the browsing the web – that is the only user-program that is visibly executing in the user-environment is the web-browser – then a number of parameters can be fixed by hard-coding them. This saves a lot of resources, especially memory and process complexity, although it does kill extensibility. Then again, if higher capabilities are needed, a move towards better hardware is the most obvious choice for the applications of this system. An operating system, in which the only the user-program is the web-browser can thus be effectively used as a web-workstation, the central objective of this project.

## 2) The operating system to be chosen as the model of development:

### Analysis:

In the present context the choices for the model operating system are as follows: -

- MS-DOS
- UNIX SVR4
- Linux
- Other operating systems

Following is a (not remotely exhaustive) list of advantages and disadvantages of using either of the above operating system as a model for development:

### *MS-DOS*

#### *Advantages:*

1. Small in size. Portable to many machines.
2. Consistent user-interface
3. Based on FAT-based file-system which is easier to implement.

#### *Disadvantages:*

1. Source code both copyrighted and unavailable
2. Not multiprogramming
3. Total addressable memory size small ( $\leq 1$  MB)
4. Multitasking possible only with the help of interrupt handling which the user-programs must implement

### *UNIX System V Release 4 (SVR4)*

#### *Advantages:*

1. Multiprogramming (time-sharing)
2. Multi-user operating system
3. Source-code of variants (Free BSD, Linux, Minix, etc.) available
4. Extensive documentation available

---

<sup>1</sup> Figures obtained from support.microsoft.com

5. Network support inherent
6. Provides a consistent interface for devices and files

*Disadvantages:*

1. Unix kernel is not multi-threaded, and so is not light-weight
2. Unix is based on the client/server model which is not always the best design for services.
3. Unix kernel is monolithic. This means that the functionality of the kernel cannot be changed without completely re-compiling the kernel.

*LINUX*

*Advantages:*

1. Open source and actively documented.
2. Multi-threaded kernel
3. Multiprogramming and multi-user
4. Inherits consistency from Unix
5. Allows module loading, thus providing most of the advantages of micro kernels.

*Disadvantages:*

1. Difficult to implement components – filesystem, kernel, drivers.
2. Linux kernels demand POSIX compliance which, bearing in mind the time constraints, is infeasible to achieve.

*Comparison:*

The following table compares the various features to be considered for the selection of the model operating system:

	MS-DOS	Unix	Linux
1.) Features			
Multiprogramming	No	Yes	Yes
Multithreading	No	No	Yes
User-interface	CUI	CUI/GUI	CUI/GUI
2.) Hardware Req.			
RAM	< 1 MB	2 MB	2-4 MB
HDD	< 5 MB	30 – 40 MB	10 – 20 MB
3.) Complexity			
Drivers	Low	Higher	Highest
Filesystem	FAT 16/32 (Easy)	NFS (Difficult)	EXT2/3 (Very difficult)
Kernel	Single-process	Multiprocessing	Multi-threaded

Thus, owing to the time constraints and the complexity requirements and features of other operating systems, UNIX is chosen as the model upon which the present project will be based.

*3.) What all software to develop:*

With respect to what all software components will be needed to develop, there are three possible approaches to developing this solution.

*Approach 1:* Develop the kernel and use an existing browser as a component.

*Approach 2:* Develop the browser with an existing kernel as the base.

*Approach 3:* Develop both the browser and the kernel.

*Approach 1:*

- The following software components will be needed to be developed:
  - ♣ Kernel – Process manager, I/O management
  - ♣ File system
  - ♣ Device Drivers
  - ♣ Network-subsystem – Protocol stacks, NIC or modem drivers
  - ♣ System calls library
  - ♣ Boot loader and program loader
- The following risks maybe associated with this approach:
  - ♣ A compatible web-browser may not be available
  - ♣ The source code of the web-browsers may not be available for any modifications
  - ♣ The browser itself may be a performance bottle-neck.

*Approach 2:*

- The following software components will be needed to be developed:
  - ♣ HTTP clients
  - ♣ HTML parsers and DTD builders
  - ♣ Layout/Rendering engines
  - ♣ Browser-control managers
  - ♣ DNS client
  - ♣ Cache manager

*Approach 3:*

- All software components from the above two approaches need to be developed.
- All compatibility issues are addressed in a positive manner.
- The time constraints might not permit this approach.

*Chosen Approach:*

The approach chosen for now is basically that of the approach number 2. However, the browser is modeled as a simplistic, text-based web-browser. The operating system services are to be provided by the Linux kernel (v2.7). The web-browser behaves as a shell for the operating system. Thus, as soon as the system boots up and loads the shell, the web-browser takes over the controls.