New Synchronizer

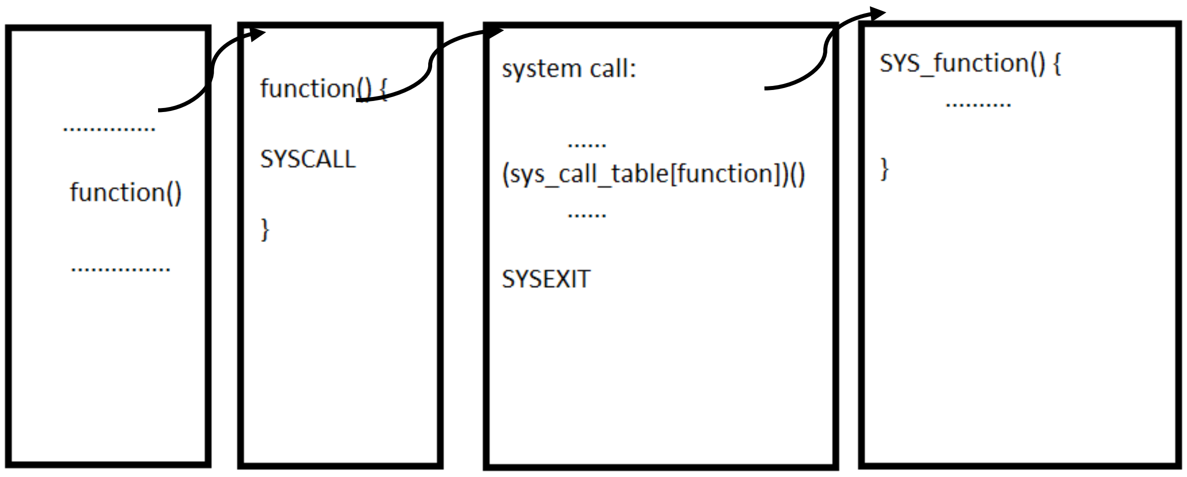
Aditya Parikh – 121001
Atman Jain – 121007
Erali Shah - 121012

Aim:

To create a new synchronizer which will allow users to create new system call to access their new events. Create different data structures based upon different system calls. Also design and create a single text-based video producer and video consumers to use the events.

What is System calls?

System calls provide a means for user or application programs to call upon the services of the operating system. System call in UNIX are mostly executed at kernel level, although there is no need of context switching. Generally written in C or C++, although some are written in assembly for optimal performance. Four type of model can be created: Many to one, one to many, many to many, hybrid. System call is called from user mode and routine is executed from kernel mode. Easiest way to implement system calls is using interrupts. A system cannot directly be called from user mode, rather we have to look in the interrupt table if available or create new one into the interrupt table. Generally operating system follows system call tables to save every system call routines uniquely.



TO DO:

When a new event is created, a new system call is added to the table which stores all the system calls uniquely. Also we need to generate a system call “Stub” which will allow our ordinary user program to invoke our system call. We need to define privileges for different system calls as operating system executes at the highest level of privileges and allows applications to call system calls. We need to create one header file and one .c file. Header file will define the system call while .c file will include the header file into it and call the system call from it. There is also one main program which to be included into init function which will be able to initialize during booting. Also we need to define every data structures related to every system calls and all the parameters to be passed or fetched during the system call. System call is never called from user mode. So we need to define another function which will invoke the procedure. Also on kernel side the table need to be invoked corresponding to the unique id of the system process. Also we need to create a new data entry into the table if the event is occurred for the first time and generate the unique id for the same. Different errors needed to be taken care of as follows:

Errors Expected to occur once the system call is called:

- Operation not permitted.
- No such file or directory.
- No such process.
- Interrupted system call.
- I/O error.
- No child process.
- Access permission denied.
- Resource temporally unavailable.

System calls can be divided into six categories as follows:

1. **Process Control**
- Process control system calls include end, abort, load, execute, create process, terminate process, get/set process attributes, wait for time or event, signal event, and allocate and free memory. Processes must be created, launched, monitored, paused, resumed, and eventually stopped. When one process pauses or stops, then another must be launched or resumed When processes stop abnormally it may be necessary to provide core dumps and/or other diagnostic or recovery tools

2. **File Management**
- File management system calls include create file, delete file, open, close, read, write, reposition, get file attributes, and set file attributes. Operations may also be supported for directories as well as ordinary files.

3. **Device Management**
- Device management system calls include request device, release device, read, write, reposition, get/set device attributes, and logically attach or detach devices. Devices may be physical, or virtual. Some systems represent devices as special files in the file system, so that accessing the "file" calls upon the appropriate device drivers in the OS.

4. **Information Maintenance**
- Information maintenance system calls include calls to get/set the time, date, system data, and process, file, or device attributes. Systems may also provide the ability to dump memory at any time, single step programs pausing execution after each instruction, and tracing the operation of programs, all of which can help to debug programs.

5. **Communication**
- Communication system calls create/delete communication connection, send/receive messages, transfer status information, and attach/detach remote devices. Communication principles can be implemented using two models: The message passing model and the shared memory model. Message passing is simpler and easier, particularly for inter-computer communications, and is generally appropriate for small amounts of data. Shared memory is faster, and is generally the better approach where large amounts of data are to be shared, particularly when most processes are reading the data rather than writing it, or at least when only one or a small number of processes need to change any given data item.

6. **Protection**
- Protection provides mechanisms for controlling which users / processes have access to which system resources. System calls allow the access mechanisms to be adjusted as needed, and

for non-privileged users to be granted elevated access permissions under carefully controlled temporary circumstances. Once only of concern on multi-user systems, protection is now important on all systems, in the age of ubiquitous network connectivity.