The Context of The Process

Himanshu Dewan (24)

Basics

- Regions
- Page and Page Table
- Memory Management Unit
- u area
- Process table entry

The Context of a Process

The context of a process consists of:

1 . Contents of its (user) address space, called as **user level context**

2 . Contents of hardware registers, called as **register context**

3 . Kernel data structures that relate to the process, called as **system context**

User Level Context

User level context consists of :

- The **process text, data, user stack, and shared memory** that occupy the virtual address space of the process.

- Parts of the virtual address space of a process that periodically do not reside in main memory because of swapping or paging.

REGISTER CONTEXT

- **Program counter** specifies the next instruction to be executed. It is an address in kernel or in user address space.
- The **processor status register (PS)** specifies hardware status relating the process. It has subfields which specify if last instruction overflowed, or resulted in 0, positive or negative value, etc. It also specifies the current processor execution level and current and most recent modes of execution (such as kernel, user).
- The **stack pointer** points to the current address of the next entry in the kernel or user stack. If it will point to next free entry or last used entry it dependent on the machine architecture. The direction of the growth of stack (toward numerically higher or lower addresses) also depend on machine architecture.
- The **general purpose registers** contain data generated by the process during its execution

SYSTEM LEVEL CONTEXT

- **static part**
- **dynamic part**

A process has one static part throughout its lifetime. But it can have a variable number of dynamic parts.

system level context : Static Part

The **static part** consists of the following components:

- The **process table entry** of a process defines the state of a process, and contains control information
- The **u area** of a process contains process control information that need be accessed only in the context of the process.
- **Pregion entries, region tables and page tables**, define the mapping from virtual to physical addresses and therefore define the text, data, stack, and other regions of a process.

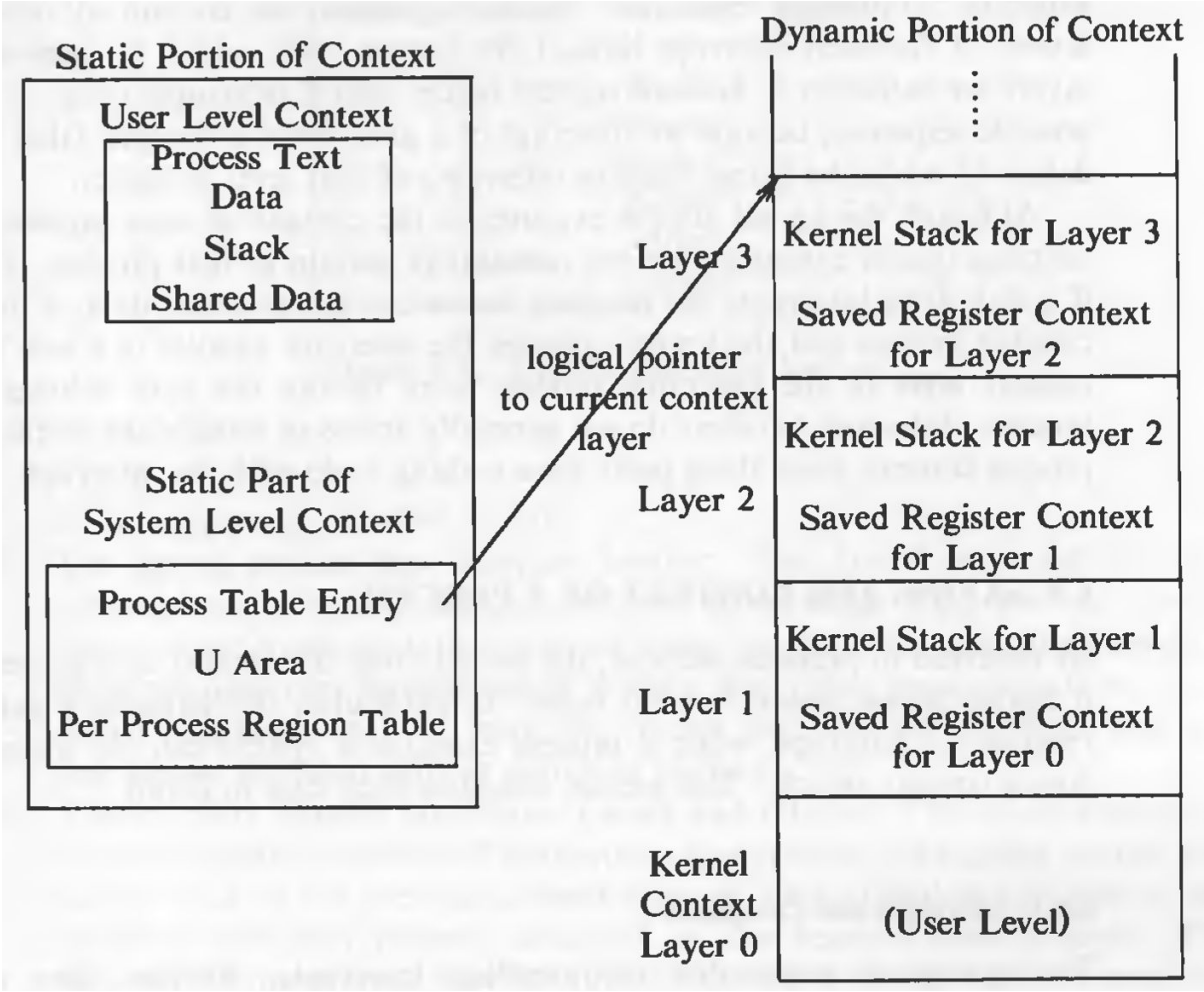
system level context : Dynamic Part

The Dynamic part consists of the following components:

- **The kernel stack contains the stack frames the kernel functions.** Even if all processes share the kernel text and data, kernel stack needs to be different for all processes as every process might be in a different state depending on the system calls it executes. The pointer to the kernel stack is usually stored in the u-area but it differs according to system implementations. The kernel stack is empty when the process executes in user mode
- The dynamic part of the system level context consists of a **set of layers**, visualized as a **last-in-first-out stack**. Each system-level context layer contains information necessary to recover the previous layer, including register context of the previous layer.

Kernel Stack Layers

The kernel pushes a context layer when an interrupt occurs, when a process makes a system call, or when a process does a context switch. It pops a context layer when it returns from an interrupt handler, returns from a system call, or when a context switch happens. Thus, a context switch entails a push-pop operation..



Source: The Design of the UNIX Operating System, by Maurice J. Bach (Ch 6.3)

dummy layer that represents the user-level context

Figure Explanation

The right side of the figure shows the dynamic portion of the context. It consists of several stack frames where each stack frame contains saved register context of the previous layer and the kernel stack as it executes in that layer. System context layer 0 is a dummy layer that represents the user-level context . .

The maximum number of context layer depends on the number of levels of interrupts the system supports.

Suppose the system supports 5 levels interrupts (software, terminal, disk, peripheral devices, clock), then 7 context layers are enough. 1 for user-level context, 1 for system call and 5 for different interrupt levels. As the kernel blocks all the lower level interrupts when a higher level interrupt occurs, there could be maximum 1 interrupt of each level. Therefore, in the worst case (interrupts occurring in the sequence of level 1 to 5), 7 context layers will be used.

Appendix

Two kernel data structures describe the state of a process: the process table entry and the u-area. The process table contains information that should be accessible to the kernel and the u-area contains the information that should be accessible to the process only when its running.

The fields in the process table are the following:

- State of the process
- Fields that allow the kernel to locate the process and its u-area in main memory or in secondary storage. This information is used to do a *context switch* to the process when the process moves from state *ready to run in memory* to the state *kernel running* or from the state *preempted* to the state *user running* or when *swapping* the process. It contains a field that gives the size of the process so that the kernel knows how much space to allocate for the process.
- Several user identifiers (user IDs or PIDs) specify the relationship of processes to each other. These ID fields are set up when the process enters the state *created* in the *fork* system call.
- Event descriptor when the process is *sleep*ing.
- Scheduling parameters allow the kernel to determine the order in which processes move to the states *kernel running* and *user running*.
- A signal fields enumerates the signals sent to a process but not yet handled.
- Various timers give process execution time and kernel resource utilization. These are used for calculation of process scheduling priority. One field is a user-set timer used to send an alarm signal to a process.

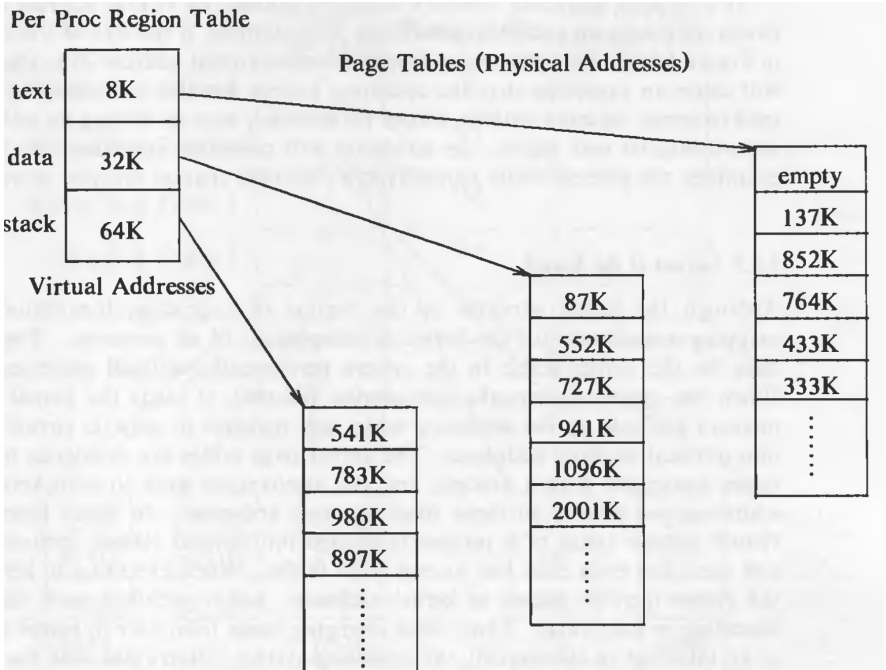
Appendix

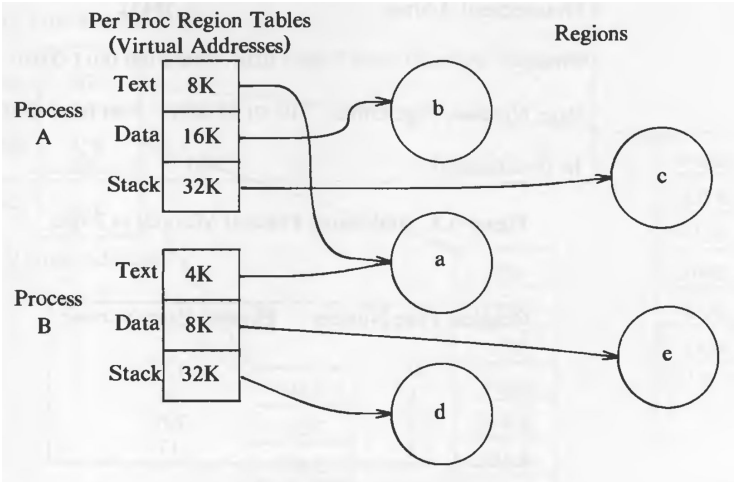
The u-area contains these fields (some are covered previously as well) :

- A pointer in the process table identifies the entry that corresponds to the u-area.
- The real and effective user IDs determine various privileges allowed the process, such as file access rights.
- Timer fields record the time the process spent executing in user mode and in kernel mode.
- An array indicates how the process wishes to react to signals.
- The control terminal field identifies the "login terminal" associated with the process, if one exists.
- An error field records errors encountered during a system call.
- A return value field contains the result of system calls.
- I/O parameters describe the amount of data to transfer, the address of the source (or target) data array in user space, file offsets for I/O, and so on.
- The current directory and current root describe the file system environment of the process.
- The user file descriptor table records the files the process has open.
- Limit fields restrict the size of a process and the size of a file it can write.
- A permission modes field masks mode settings on files the process creates.

Appendix - Region and Page Tables

The UNIX system divides its virtual address space in logically separated *regions*. The regions are contiguous area of virtual address space.





Thank You

References :

The Design of the UNIX Operating System, by Maurice J. Bach

2022 Summer Advance Operating System Course Taught by Dr.Bharti