

1 Process

Vocabularies

1. Process

- Is a program in execution

2. Running Program

- Is a collection of coded software instructions that can be executed by a computer to perform a specific task

3. Time Sharing

- Is a basic technique used by an OS to share a resource
- Allows an entity to use the resource for a little while, and then a little while by another, and so forth

Example

CPU

4. Space Sharing

- Is where a resource (space) is divided among those who wishes to use it

Example

Disk, and Memory

5. Mechanism

- Is a low-level method or protocol that implement a needed piece of functionality.

Example

Context Switching

6. Policy

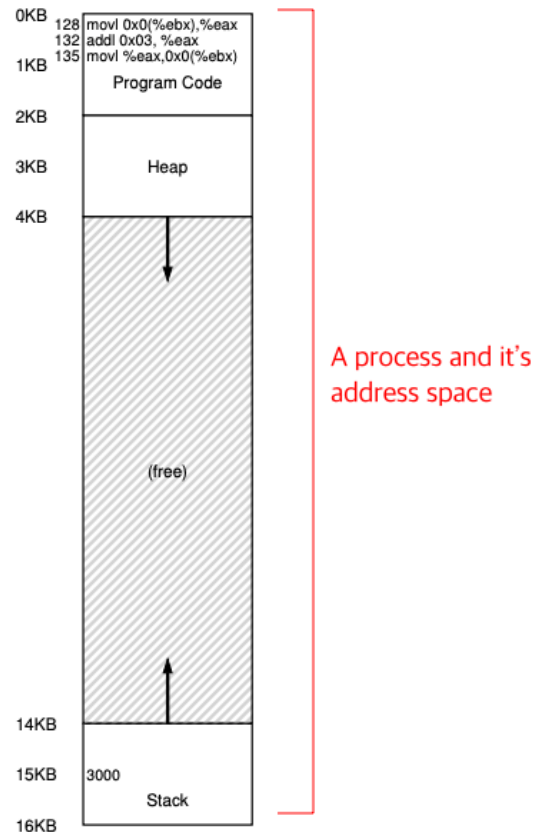
- Is an algorithm for making some kinds of decision within the OS

Example

Scheduling Policy. That is, what kind of program should the OS run?

7. Address Space

- Is a range of discrete addresses where each corresponds to a memory cell

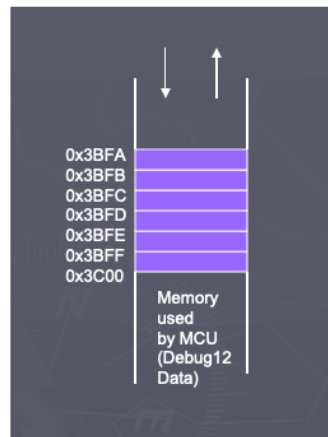


8. Program Counter

- Is also called **Instruction Pointer**
- Is a process register that tells which instruction of the program is currently being executed

9. Stack Pointer

- Is a register that points to the location of last item placed in memory block



10. Frame Pointer

- Is a reference pointer allowing a debugger to know where local variable or an argument is at with a single constant offset

```
4.c:7:6: note: expected 'struct abc *' but argument is of type 'struct abc *'
void count_length(struct abc *_el);
```

↑
Frame Pointer

11. Eager Loading Process

- Is the process that loads all code and data before running the program

12. Lazy Loading Process

- Is the process that loads piece of code or data only as they are needed during program execution

13. Stack

- Is also called **runtime stack**, **automatic memory**
- Is a special region in computer's memory that temporarily stores local variables, function parameters, and return addresses
- Is managed by compiler

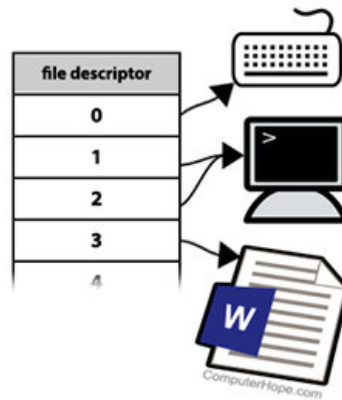
14. Heap

- Is a user-managed region in computer memory
- Is used for dynamically-allocated data structures such as linked list, hash-tables, and trees

- Is allocated using `malloc`, `calloc`, and `realloc`

15. File Descriptors

- Is a number that uniquely identifies an open file in a computer's operating system



16. Process States

- Is also called **kernel state**
- Is the state field in a **process control block**.

Example

Ready, Running, Blocked

17. Process List

- Is also called **task list**
- Contains information about all the processes running in the system
- Contains **process control block** in each entry

18. Context Switch

- is the process of storing the state of a process or thread, so that it can be restored and resume execution at a later point

19. Register Context

- Is the data structure where contents of registers are saved before a process switches into blocked state

```

// the registers xv6 will save and restore
// to stop and subsequently restart a process
struct context {
    int eip;
    int esp;
    int ebx;
    int ecx;
    int edx;
    int esi;
    int edi;
    int ebp;
};

// the different states a process can be in
enum proc_state { UNUSED, EMBRYO, SLEEPING,
    RUNNABLE, RUNNING, ZOMBIE };

// the information xv6 tracks about each process
// including its register context and state
struct proc {
    char *mem;           // Start of process memory
    uint sz;             // Size of process memory
    char *kstack;        // Bottom of kernel stack
                        // for this process
    enum proc_state state; // Process state
    int pid;             // Process ID
    struct proc *parent;  // Parent process
    void *chan;          // If !zero, sleeping on chan
    int killed;          // If !zero, has been killed
    struct file *ofile[NOFILE]; // Open files
    struct inode *cwd;    // Current directory
    struct context context; // Switch here to run process
    struct trapframe *tf; // Trap frame for the
                        // current interrupt
};

```

register context data structure

Process Control Block

Where register context is

20. Process Control Block

- Is also called **process descriptor**
- Is a data structure used by computer operating systems to store all the information about a process

21. Zombie State

- Is a process that has completed execution but still has an entry in the process table

1.1 Process

- Is named by process ID or PID
- Is comprised of
 - Address Space
 - CPU Registers
 - Program Counter
 - Stack Pointer
 - Frame Pointer
 - I/O Information

1.2 Process API

- has the following methods in any operating systems
 - **Create**
 - * Is a method for creating a new process
 - * Invoked to OS when
 - A command is typed into shell
 - An application icon is double-clicked
 - **Destroy**
 - * Is a method for forcefully destroying a process

Example

`kill`

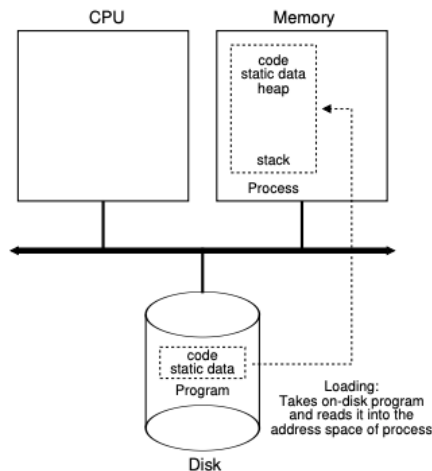
- **Wait**
 - * Is a method that causes a process stop running until a signal is given
- **Miscellaneous Control**
- **Status**
 - * Is a method for getting information about a process

Example

How long it has run for, what state it is in

1.3 Process Creation: A little more detail

- Steps
 1. Type a command into commandline / Double click an application
 2. Load program code and static data (e.g. initialized variables) into memory, into the address space of the process



- Eagerly loading process in early days
- Lazy loading process today

1.4 Process States

1.5 Data Structures