Operating System

An OS usually has two parts:

kernel code which manages the computer and allows users programs to run.

user land utilities and libraries to allow users interact with the kernel and perform basic tasks.

Programs are referred to as applications.

The kernel may provide:

- resource management (cpu, memory, disk usage),
- provide services (networking, filesystems, memory protection),
- standards (API, ABI, hardware abstraction).

From Batching to Scheduling

Batch Systems A batch system runs a single program to completion, and then begins the next job.

Batch Multiprogram Several programs are kept in memory. When one stops to do I/O (*blocks*) the next is run.

Coop Multitasking Several programs. When one gives up the CPU another is chosen to run.

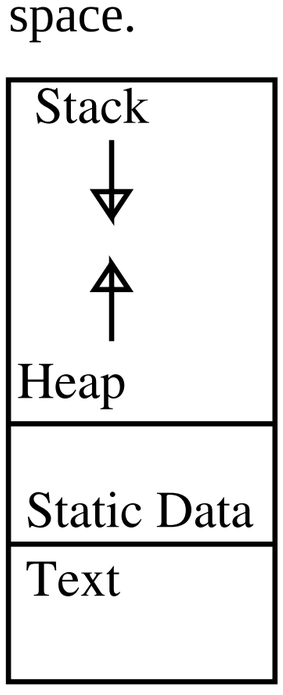
Preemptive Multitasking Several programs. When any program is started a timer is set and when the timer expires another program is selected.

The issue is now selecting the next process.

Processes & Memory

A running program is called a *process*. The switch from one process to the next is a *context switch*. The time a process can run before being preempted is it's *quantum* or *time slice*.

A process can be provided with it's own virtual address space.



Example: Show how the following might effect the processes' address space.

```
int main() {
    struct_matrix *m;

    m = alloc_matrix(10);
    fill_matrix(m,1.0);
    free_matrix(m);
}

struct matrix alloc_matrix(int size) {
    struct matrix *r;

    r = malloc(sizeof(struct matrix));
    r→size = size;
    r→data = malloc(sizeof(double)*size*size);

    return r;
}

void fill_matrix(struct matrix *m,double value) {
    int i,j;

    for( i = 0; i < m→size; i++ )
        for( j = 0; j < m→size; j++ )
            r→data[i*m→size + j] = value;
}

void free_matrix(struct matrix *m) {
    free(m→data);
    free(m);
}
```

Threads

A context switch can be expensive. To avoid context switches a process may have several *threads* of execution, which share one address space. This reduces the cost of a context switch.

Threads are used to simplify programming (eg. ‘multi-threaded server’) or to take advantage of multi-cpu computers. Threading can be done by hand, with a library or with kernel support. Various standards for threading exist (eg. POSIX threads).

Processes under Unix

You can look at the list of existing processes on Unix using `ps` or `top`. For each process the OS remembers things like process id, parent pid, child processes, limits, open files, locks, accounting data, who the process is running as...

A process can be in different states:

Running Currently being executed.

Ready Waiting for CPU.

Blocked Waiting for some resource.

Zombie Finished, but waiting for parent to collect status.

In Unix new processes are created using `fork()`, which makes an almost identical copy of the original program.

```
pid = fork();
switch(pid) {
case −1:
   printf("Fork failed.");
   break;
case 0:
   printf("I am the child.");
   break;
default:
   printf("I am parent. Child is %d.",
      pid);
   break;
}
```

The `exec()` replaces the currently running process by starting a program on disk.

```
execl("/bin/ls", "ls", "-l", "/etc",
NULL);
```

A process finishes when it calls `exit()`, or if it is sent a fatal *signal*. Signals exist for many conditions (see `man signal`).

Scheduling under Unix

There are two numbers associated with scheduling a process on Unix. The *nice* value and then *priority*. After each time a process is run it's priority value is increased according to the amount of it's quantum is used and its nice value. Processes with a low value in priority are selected for running.

Some versions of Unix provide extensions to this providing real-time scheduling and idle-time scheduling.

References

- “Operating System Concepts”, Silberschatz & Galvin, Addison-Wesley.
- “Design an Implementation of the BSD 4.4 Operating System”, Karls, McKusick,...
- `/usr/src/sys` on BSD machines and `/usr/src/linux-*` on Linux machines.
- “Advanced Programming in the Unix Environment”, Stevens, Addison-Wesley.