/proc/sys/kernel/pid_max

```
defines max pid in linux systems
```

The parent of any process can be found by looking at the Ppid field provided in the Linux-specific /proc/PID/status file

- Although not specified in SUSv3, the C program environment on most UNIX implementations (including Linux) provides three global symbols: etext, edata, and end. These symbols can be used from within a program to obtain the addresses of the next byte past, respectively, the end of the program text, the end of the initialized data segment, and the end of the uninitialized data segment. To make use of these symbols, we must explicitly declare them, as follows:

  ```
  extern char etext, edata, end;
  /* For example, &etext gives the address of the end
             of the program text / start of initialized data */
  ```

Locality of Reference

Locality of reference refers to the tendency of programs to access certain memory locations more frequently than others. This concept is essential for understanding memory optimization techniques in computer systems.

Types of Locality

1. **Spatial Locality**
   Spatial locality is the tendency of a program to reference memory addresses that are near those recently accessed. This behavior often occurs due to:
   - Sequential processing of instructions.
   - Sequential processing of data structures.

2. **Temporal Locality**
   Temporal locality is the tendency of a program to access the same memory

Resident Set

The pages of a program need to be resident in physical memory page frames; these pages form the so-called `Resident Set`

- Page Size in Unix:

```
long x = sysconf(_SC_PAGESIZE)
```

- If a process tries to access an address for which there is no corresponding page-table entry, it receives a **SIGSEGV** signal

- Where appropriate, two or more processes can share memory. The kernel makes this possible by having page-table entries in different processes refer to the same pages of RAM. Memory sharing occurs in two common circumstances:
  - Multiple processes executing the same program can share a single (read-only) copy of the program code. This type of sharing is performed implicitly when multiple programs execute the same program file (or load the same shared library).
  - Processes can use the shmget() and mmap() system calls to explicitly request sharing of memory regions with other processes. This is done for the purpose of interprocess communication.

User stack vs Kernel Stack

Sometimes, the term user stack is used to distinguish the stack we describe here from the kernel stack. The kernel stack is a per-process memory region maintained in kernel memory that is used as the stack for execution of the functions called internally during the execution of a system call. (The kernel can’t employ the user stack for this purpose since it resides in unprotected user memory.)

Each (user) stack frame contains the following information:

- **Function arguments and local variables:** In C these are referred to as automatic variables, since they are automatically created when a function is called. These variables also automatically disappear when the function returns (since the stack frame disappears), and this forms the primary semantic distinction between automatic and static (and global) variables: the latter have a permanent existence independent of the execution of functions.
- **Call linkage information:** Each function uses certain CPU registers, such as the program counter, which points to the next machine-language instruction to be executed. Each time one function calls another, a copy of these registers is saved in the called function’s stack frame so that when the function returns, the appropriate register values can be restored for the calling function.

Why argv[0] ?

The fact that argv[0] contains the name used to invoke the program can be employed to perform a useful trick. We can create multiple links to (i.e., names for) the same program, and then have the program look at argv[0] and take different actions depending on the name used to invoke it. An example of this technique is provided by the gzip(1), gunzip(1), and zcat(1) commands, all of which are links to the same executable file. (If we employ this technique, we must be careful to handle the possibility that the user might invoke the program via a link with a name other than any of those that we expect.)