Errors

Common to all system services is that they all return either a valid response to the request, or a value that indicates an error situation.

Often a return value of -1 indicates that something went wrong, while 0 or some positive value indicates success. There are exceptions to this rule, so read the manual page for the service. The error return is however always some value that is not a legal value for the type of object being returned.

It is often not enough to know that "an error occurred". We usually like to know the type of error or the reason it occurred so that we can try to correct it (if possible). Not all errors can be dealt with, but some are less serious than others.

We have here a conflict of interest: on the one hand we do not want to reserve too many values for error returns, but on the other hand we want as much information as possible about the nature of the error.

This is solved by setting a global variable called `errno` in the user process with a value before returning from system calls. Normally `errno` has the value of 0, indicating that no error has occurred. When a system call returns -1, this indicates that we should check the value of `errno` to determine the nature of the error.

On SunOS there are more nearly 100 different error situations that can occur. Many of these are uncommon, but for every system call there are usually 5 or 6 errors that are likely. Again, the manual page for each system call describes the errors that can apply.

perror

Note that all of these error values are just numbers – when we get -1 from a system call we know that an error has occurred. We then check `errno` to get the corresponding error number and we can take appropriate action.

But often we want to present a message for the user to indicate what has happened, and so there is also a short descriptive text available.

`perror` is a useful function that will display the text associated with the current value of `errno`. `perror` also gives us the opportunity to print a short text *before* the error message, indicating for example where in our code the error occurred.

For example:

```
p=fork();

switch (p) {
case -1:
  perror("fork failure");
  exit(1);

case 0:

default:

}
```

It is important when an error has occurred to call `perror` as soon as possible, and in all cases before a new system call (or library function that uses system calls), because *all* system calls can change the value of `errno`.