Unix/Linux Exit Codes and Customization Cheat Sheet

Standard Unix/Linux Exit Codes

0 - Success

1 - General Error

2 - Misuse of Shell Builtins

126 - Command Invoked but Not Executable

127 - Command Not Found

128 - Invalid Exit Argument

130 - Script Terminated by Ctrl+C (SIGINT)

137 - Process Killed (SIGKILL)

139 - Segmentation Fault

143 - Terminated by SIGTERM

255 - Exit Status Out of Range

Custom Exit Codes in C (Example)

```
#define FILE_NOT_FOUND 1
#define INVALID_INPUT 2
#define CALCULATION_ERROR 3
#define MEMORY_ERROR 4
```

Example Usage:

```
if (file == NULL) {
    return FILE_NOT_FOUND;
}

if (input < 0) {
    return INVALID_INPUT;
}
```

Unix/Linux Exit Codes and Customization Cheat Sheet

Notes

- Use 0 for successful program completion.
- Use different non-zero codes to represent different types of errors.
- exit() can be used anywhere in the program to stop immediately with a specific code.
- Shell: Use `echo $?` to see the last program's exit code.
- Exit codes above 128 often mean system signals (like kill, Ctrl+C).