

Chapter 1 : Introduction

* The difference between ANSI-C and K&N-C

(1) Function Prototyping: ANSI-C supports strong type checking, it enables the compiler to check for the function calls in the user prg that passes invalid no of arg, return type of the function. Whereas K&R-C compiler won't fix this error and it may lead to prg crash when they are executed.

(2) Support of constant and volatile keyword:
Constant keyword declares that some data cannot be changed. Volatile keyword specifies that value of some variables may change asynchronously. Not Supported in RfR-C.

Ex: Volatile char * port = 0x7777;

```
char *ch = "port";
```

```
ch = *port;
```

LC-TYPE

LC - TIME

LC - NUMERIC.

(3) Supports ^{narrow}void character and Internationalisation
^{narrow}void character uses more than one byte of storage per character. ANSI-C support setlocale function which allows users to specify the format of date and set nos in different countries.

char, setlocate, int category, const char, &locate
 ↳ date, number

(4) Permits function pointers to be used ^{it} without ^{count} referencing

void foo(double xyz, char *temp);

void (*funpt)(double, char *) = foo;

ANSI-C { foo(12.3, "Hi");
funpt(12.3, "Hi");

K&R { (*funpt)(12.3, "Hi");

ANSI-C specifies that a function ptr may be used as a function name. No dereferencing is need when calling a function whose addr is contained in the pointer.

* Difference Between ANSI-C and C++

ANSI-C

(1) It uses K&R-C defaults. Function declaration for any function that are referenced before their declaration in the prog is accepted.

(2) void foo();
its equivalent in ANSI-C is

void foo(...): This means the funⁿ 'foo' can be called with any no. of actual arguments.

(3) It does not employ type safe linkage.
↳ invalid declared referenced

C++

It requires that all functions must be declared or defined that can be referenced.

void foo(); its equivalent in C++ is:
void foo(void);

means that funⁿ 'foo' may not accept any arguments

It enforces external function names for type safe linkage.

* ANSI-C also defines a set of C-preprocessor symbols which may be used in the user program and are assigned values at compile time.

The various symbols are:

- **STDC** - : This macro is used as a test macro, that is, if its value is '1', it is a ANSI-C compiler if '0', some other compiler.

- **__LINE__** - : It displays the line number of a source file for which the symbol is referenced.

- **__FILE__** - : It displays the file name that contains this symbol.

- **__DATE__** - : It specifies the date of when the file was compiled.

* POSIX STANDARDS :

Many versions of UNIX exist today and each of them provide its own set of API's, it is difficult for system developers to create the applications that can be ^{easily} ported on different version of UNIX.

To overcome this the IEEE society found the special task force called POSIX to create a set of standards for operating system interface.

POSIX.1

This committee proposed a set of standards for base OS API's which specifies for manipulation of files and processes.

POSIX.1b

This committee proposed a set of standard API's for real time OS which included IPC.

POSIX.1c.

This standard specifies multithreaded programming interface.

* POSIX's TEST MACROS:

- ✓ **POSIX_JOB_CONTROL** : This macro supports the Berkeley Style Distribution.
BSD style job control.
- ✓ **POSIX_SAVED-ID** : Here each process running on the system keeps the set-UID and set-GID so that it can change the effective UID and GID to those values via set-UID and set-GID.
- ✓ **POSIX_CHOWN_RESTRICTED** : If this macro value is -1 it changes the ownership of the file otherwise only privileged user may change the ownership of the file.
- ✓ **POSIX_NO_TRUNC** : If the macro value is -1 any long path name passed to it is truncated to name_max (NAME_MAX) bytes otherwise an error is generated.
- ✓ **POSIX_VDISABLE** : If the macro value is -1, there is no disabling characters for special characters for all terminal device files.

* **LIMIT CHECKING AT COMPILE TIME AND AT RUN TIME**: To find actual implementation configuration we can use `sysconf`, `pathconf` and `fpathconf` functions at run-time.

200112L
103+7com
united.h

sysconf : This is used to query the system-wide configuration limits that are implemented on a given system

pathconf : It is used to query file related configuration limits where it take file path name as argument

fpathconf : It is used to query file related configuration limits where it take file descriptors as its arguments.

#include <unistd.h>

long sysconf(const int limit_name);

long pathconf(const char *pathname,
const int limit_name);

long fpathconf(const int fdes, const
int limit_name);

* The following is a list of POSIX.1-defined constants in the <limits.h> header:

Compile time limit	Min. Value	Meaning
_POSIX_CHILD_MAX	6	Max. no. of child processes that may be created at any one time by a process
_POSIX_OPEN_MAX	16	Max. no. of files that may be opened simultaneously by a process.
_POSIX_PATH_MAX	255	Max no. of characters allowed in a file name

<code>_POSIX_NAME_MAX</code>	14	Max. no of characters allowed in a file name.
<code>_POSIX_LINK_MAX</code>	8	Max no. of links a file may have.
<code>_POSIX_STREAM_MAX</code>	8	Max. no of I/O streams that may be opened simultaneously by a process

* The followed is a list of POSIX.1b - defined constants :

Compile time limit	Min. Value	Meaning
<code>_POSIX_AIO_MAX</code>	1	No. of simultaneously asynchronous I/O
<code>_POSIX_AIO_LISTIO_MAX</code>	2	Max no. of operations in one listio
<code>_POSIX_TIMER_MAX</code>	32	Max no. of timers that can be used simultaneously by a process.
<code>_POSIX_DELAYTIME_MAX</code>	32	Max no of overruns allowed per timer
<code>_POSIX_RTSIG_MAX</code>	8	Max no. of real time signals.

* FIPS (Standards) :

Federal Information Processing Standards

It is a guideline for standards. These guidelines have been extracted from POSIX.1 standards. If the system satisfies the

following features then it is said to be implemented with FIPS Standards:

1. It should support Job Control
2. It should support `set-UID` and `set-GID` functions.
3. It should not support long pathnames
4. `_POSIX_CHOWN_RESTRICTED` must be defined explicitly.
5. `_POSIX_VDISABLE` must be defined.
6. `READ` and `WRITE` API should return the no. of bytes that have been texted after API has been interrupted by signals.
7. `GID` of newly created file must inherit the `GID` of its containing directory:

* API Characteristics:

Most of the API's return an integer value which indicates the termination status of their execution. If an API return -1, it means that the API execution has failed and a global variable "`errno`" is set with an error code. The variable `perror` displays the message of the error code to the standard output or the log files.

The various error code status are:

`EACCESS` : A process does not have access permission to perform an operation.

`EPERM` : It means an API was aborted as the calling process does not ^{have} the superuser privileges.

BADF : It means an API was called with an invalid file descriptor.

ENOENT : It means an invalid filename was specified to an API.

EINTR : Which means an API execution was aborted due to signal interruption.

EAGAIN : This means an API was aborted because some of the system resources requested were temporarily unavailable.

ENOMEM : Which means the API was aborted because it could not allocate dynamic memory.

EIO : An I/O error encountered in an API execution.

ECHILD : Which means a process does not have any child process which it is waiting for.