A **file descriptor** is a low-level integer value used by operating systems to reference **open files or input/output resources** like:

- Regular files (e.g., document.txt)
- Pipes
- Sockets
- Devices (e.g., /dev/null)
- Terminals

🧠 **Key Concepts:**

- In Unix-like systems (macOS, Linux), **each process** gets a **file descriptor table**, where:
  - File descriptor **0** = Standard Input (stdin)
  - File descriptor **1** = Standard Output (stdout)
  - File descriptor **2** = Standard Error (stderr)
  - When you open a file, the OS returns a file descriptor — an integer (e.g., 3, 4, etc.) — to use for reading, writing, or other operations

Write system call:

```
#include<unistd.h>
int main()
{
 Write(1,"Hello",5);
}
```

Read and Write:

```
#include<unistd.h>
int main(){
char b[30];
int n;
n=read(0,b,30);
write(1,b,n);
}
```

Open System call:

Flag	Meaning
O_RDONLY	Open for reading only
O_WRONLY	Open for writing only
O_RDWR	Open for both reading and writing
O_CREAT	Create file if it doesn't exist
O_APPEND	Write data at the end of the file
O_TRUNC	Truncate (empty) file if it exists
O_EXCL	Fail if file already exists (used with O_CREAT)

```
#include <fcntl.h>    // For open() and flags like O_RDONLY, O_CREAT, etc.
#include <unistd.h>   // For close(), read(), write(), etc.
#include <sys/types.h> // (Optional) For data types like mode_t
#include <sys/stat.h>  // For file permission flags (used with O_CREAT)
```

The three user categories:

1. **Owner** (user who owns the file)
2. **Group** (members of the file's group)
3. **Others** (everyone else)

Permission Values:

- **4** = Read (r)
- **2** = Write (w)
- **1** = Execute (x)

```
//
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
   int n, fd;
   char buf[50];
   fd = open("test.txt", O_RDONLY);
   n = read(fd, buf, 10);
   write(1, buf, 10);
   close(fd);
   return 0;
}

//
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
   int n, fd, fd1;
   char buf[50];
   fd = open("test.txt", O_RDONLY);
   n = read(fd, buf, 50);
   fd1 = open("target", O_CREAT | O_WRONLY, 0642);
   write(fd1, buf, n);
```

```
    close(fd);
    close(fd1);
    return 0;
}

//

#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
    int n, fd1;
    char buf[50];
    n = read(0, buf, 20); // Read from stdin (file descriptor 0)
    fd1 = open("target", O_WRONLY);
    write(fd1, buf, n);   // Fixed typo: was 'bufn'
    close(fd1);
    return 0;
}

//

#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

int main() {
    int n, fd1;
    char buf[50];

    n = read(0, buf, 20); // Read 20 bytes from stdin (file descriptor 0)
    fd1 = open("target", O_WRONLY | O_APPEND); // Open file in append mode
    write(fd1, buf, n); // Write the read bytes to the file
    close(fd1); // Close the file descriptor

    return 0;
}
```

Seeking:

Value	Meaning
SEEK_SET	From the beginning of the file
SEEK_CUR	From the current position

SEEK_END	From the end of the file

lseek(fd, 0, SEEK_SET); // Go to the beginning of the file
lseek(fd, 10, SEEK_CUR); // Skip 10 bytes ahead from current position
lseek(fd, -5, SEEK_END); // Go 5 bytes back from the end

Dup:
int dup(int oldfd);

🔁 **Also Related: dup2 and dup3**

- dup2(oldfd, newfd) → duplicates oldfd into newfd (closes newfd first if needed)
- dup3(oldfd, newfd, flags) → same as dup2 but with extra options (Linux-specific)

//
Here's a **well-organized table** of **basic shell scripting functions/features**, followed by **15 small example scripts** for common beginner-friendly tasks like even/odd, factorial, sum, etc.

Shell Scripting Basics Table

Concept Syntax/Example
Printecho "Hello World"
Read input read name
Variable x=5, name="Alice"
Arithmetic sum=$((a + b))
If condition if [$x -gt 0]; then ... fi
While loop while [$x -gt 0]; do ... done
For loop for i in 1 2 3; do ... done
Until loopuntil [$x -eq 0]; do ... done
Function function greet() { echo "Hi $1"; }
Command Substitution today=$(date)
File existence if [-f file.txt]; then echo "Exists"; fi
Directory check if [-d folder]; then echo "Directory exists"; fi
Exit script exit 0
Comments # This is a comment
String comparison if ["$a" == "$b"]; then ... fi

✅ **15 Small Shell Script Examples**

1. Even or Odd

```
read -p "Enter number: " num
if [ $((num % 2)) -eq 0 ]; then
  echo "Even"
else
  echo "Odd"
fi
```

———

2. Factorial

```
read -p "Enter number: " num
fact=1
while [ $num -gt 0 ]; do
  fact=$((fact * num))
  num=$((num - 1))
done
echo "Factorial is $fact"
```

———

3. Sum of N numbers

```
read -p "Enter N: " n
sum=0
for ((i=1; i<=n; i++)); do
  sum=$((sum + i))
done
echo "Sum: $sum"
```

———

4. Check Prime

```
read -p "Enter number: " num
count=0
for ((i=2; i<num; i++)); do
  if [ $((num % i)) -eq 0 ]; then
    count=1
    break
  fi
done
[ $count -eq 0 ] && echo "Prime" || echo "Not Prime"
```

———

5. Reverse a Number

```
read -p "Enter number: " num
rev=0
while [ $num -gt 0 ]; do
  rem=$((num % 10))
  rev=$((rev * 10 + rem))
  num=$((num / 10))
done
echo "Reversed number: $rev"
```

———

6. Palindrome Check

```
read -p "Enter number: " num
org=$num
rev=0
while [ $num -gt 0 ]; do
  rem=$((num % 10))
  rev=$((rev * 10 + rem))
  num=$((num / 10))
done
[ $org -eq $rev ] && echo "Palindrome" || echo "Not Palindrome"
```

———

7. Swap Two Numbers

```
read -p "A: " a
read -p "B: " b
temp=$a
a=$b
b=$temp
echo "After swap: A=$a, B=$b"
```

———

8. Fibonacci Series

```
read -p "How many terms? " n
a=0
```

```
b=1
for ((i=0; i<n; i++)); do
  echo -n "$a "
  fn=$((a + b))
  a=$b
  b=$fn
done
echo
```

———

9. GCD (HCF)

```
read -p "Enter two numbers: " a b
while [ $a -ne $b ]; do
  if [ $a -gt $b ]; then
    a=$((a - b))
  else
    b=$((b - a))
  fi
done
echo "GCD is $a"
```

———

10. LCM

```
read -p "Enter two numbers: " a b
x=$a
y=$b
while [ $a -ne $b ]; do
  if [ $a -lt $b ]; then
    a=$((a + x))
  else
    b=$((b + y))
  fi
done
echo "LCM is $a"
```

———

11. Print Multiplication Table

```
read -p "Enter number: " n
for ((i=1; i<=10; i++)); do
```

```
  echo "$n x $i = $((n * i))"
done
```

---

12. Check Leap Year

```
read -p "Enter year: " year
if (( (year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )); then
  echo "Leap Year"
else
  echo "Not a Leap Year"
fi
```

---

13. Count Digits

```
read -p "Enter number: " num
count=0
while [ $num -gt 0 ]; do
  num=$((num / 10))
  ((count++))
done
echo "Digits: $count"
```

---

14. Sum of Digits

```
read -p "Enter number: " num
sum=0
while [ $num -gt 0 ]; do
  sum=$((sum + num % 10))
  num=$((num / 10))
done
echo "Sum of digits: $sum"
```

Command	Description	Example	
Create File	touch	Creates an empty file.	touch file.txt
	echo "text" > file	Creates a file with content.	echo "Hello" > hello.txt

Create Directory	mkdir	Creates a	mkdir my_folder
	mkdir -p	Creates nested directories.	mkdir -p dir1/dir2/dir3
Move/Rename	mv	Moves or renames files/directories.	mv old.txt new.txt
			mv file.txt /target/dir/
Copy	cp	Copies files.	cp file.txt backup.txt
	cp -r	Copies directories recursively.	cp -r folder/ backup/
Delete File	rm	Removes files.	rm file.txt
	rm -f	Forces deletion without confirmation.	rm -f file.txt
Delete Directory	rmdir	Removes **empty** directories.	rmdir empty_folder
	rm -r	Recursively deletes directories and contents.	rm -r folder/
	rm -rf	**Force** deletion (dangerous, no warnings).	rm -rf folder/
List Files	ls	Lists directory contents.	ls -l (detailed list)
View File	cat	Displays file content.	cat file.txt
	less / more	Views large files page-by-page.	less large_file.log
Find Files	find	Searches for files/directories.	find /home -name "*.txt"
Change Permissions	chmod	Modifies file permissions.	chmod 755 script.sh
Change Owner	chown	Changes file/directory ownership.	sudo chown user:group file.txt