BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY (BiT)

FACULTY OF COMPUTING

OPERATING SYSTEM INDIVIDUAL ASSIGNMENT

NAME: ABIY TILAHUN

ID : 1600589

Section: B

Submission date:16/08/2017E.C

Submitted to: Mr wondimu

System calls

Wait()

The wait() system call is an essential component of process management in Unix-like operating systems, including Netrunner OS, which is based on Debian and uses the Linux kernel. It allows a parent process to pause its execution until one of its child processes has finished executing. This system call plays a crucial role in maintaining process hierarchy, resource management, and system stability.

In a multitasking operating system like Netrunner OS, processes often create child processes using the fork() system call. Once created, these child processes run concurrently with their parent process. However, it is important for the parent process to keep track of its children, especially when they terminate. This is where the wait() system call becomes useful—it enables the parent to "wait" for the child to finish, retrieve its exit status, and clean up resources allocated to it.

Syntax

The basic syntax of wait() is:

pid_t wait(int *status);

status: A pointer to an integer where the exit status of the terminated child process will be stored.

Returns the process ID of the terminated child or -1 on error.

Functionality

When a parent process calls wait(), it is blocked until any one of its child processes exits. When the child terminates, the wait() call:

1. Removes the child from the process table (thus avoiding a zombie process).

2. Stores the exit status in the integer pointed to by status.

Using macros like WIFEXITED(status) and WEXITSTATUS(status), the parent can determine if the child exited normally and what its return value was.

Importance in Netrunner OS

Even though Netrunner OS is known for its elegant desktop environment (KDE Plasma) and user-friendliness, it still inherits the full capabilities of Linux process management under the hood. Developers and advanced users writing scripts, C programs, or managing background services can use wait() to:

Handle background jobs.

Ensure correct execution sequence of processes.
Avoid resource leaks and process table overflows due to zombie processes.
For example, if a user writes a custom script or application that launches several subprocesses (like a background backup or installation task), the use of wait() ensures that each process finishes cleanly before the parent continues or exits.

Practical Use Case

Here's a small example of how wait() might be used in a C program on Netrunner OS:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
pid_t pid = fork();

if (pid == 0) {
// Child process
printf("Child process running\n");
return 0;
} else {
// Parent process
```

```
int status;
wait(&status);
printf("Child has finished\n");
}

return 0;
}
```

Conclusion

The wait() system call is a fundamental tool in Netrunner OS for process control. While invisible to most users, it ensures that applications and services handle process lifecycles efficiently. Whether managing system processes, writing custom software, or debugging, wait() is vital for clean and predictable process behavior.