



Artificial Intelligence and Data Science Department.

OS / Even Sem 2021-22 / Mock Practicals.

YASH SARANG.

47 / D6AD.

MOCK PRACTICALS.

Aim: Write shell scripts to explore wait and waitpid before termination of process.

Theory: The waitpid() system call suspends execution of the current process until a child specified by the pid argument has changed state.
By default, waitpid() waits only for terminated children, but this behaviour is modifiable via the options argument.

The wait() system call suspends execution of the current process until one of its child terminates.

Code:

```
$ cat 47YashSarang.c
#define _POSIX_SOURCE
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

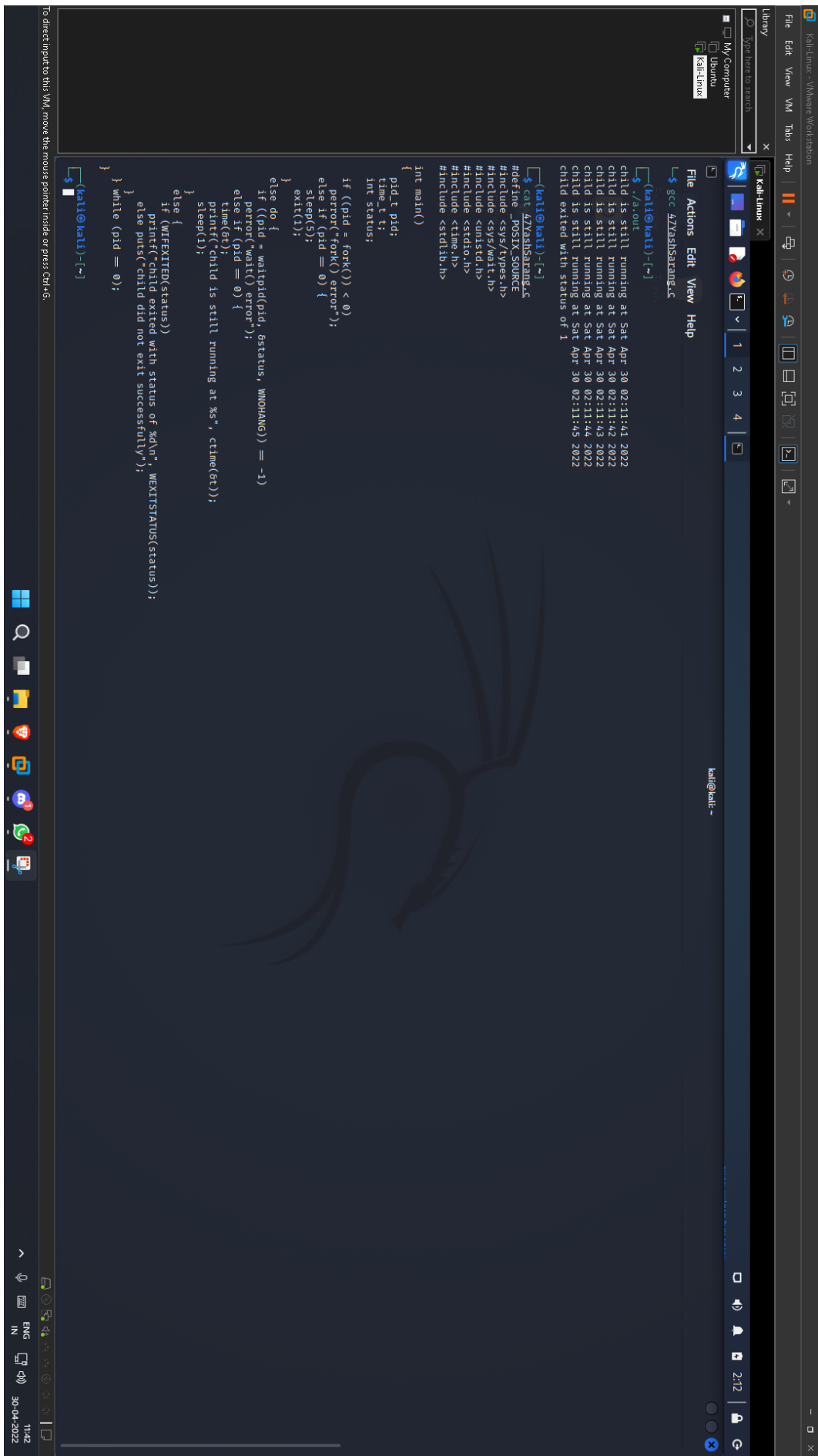
int main()
{
    pid_t pid;
    time_t t;
    int status;

    if ((pid = fork()) < 0)
        perror("fork() error");
    else if (pid == 0) {
        sleep(5);
        exit(1);
    }
    else do {
        if ((pid = waitpid(pid, &status, WNOHANG)) == -1)
            perror("wait() error");
        else if (pid == 0) {
            time(&t);
            printf("child is still running at %s", ctime(&t));
            sleep(1);
        }
        else {
            if (WIFEXITED(status))
                printf("child exited with status of %d\n", WEXITSTATUS(status));
            else puts("child did not exit successfully");
        }
    } while (pid == 0);
}
```

Output:

```
(kali㉿kali)-[~]
$ ./a.out
child is still running at Sat Apr 30 02:11:41 2022
child is still running at Sat Apr 30 02:11:42 2022
child is still running at Sat Apr 30 02:11:43 2022
child is still running at Sat Apr 30 02:11:44 2022
child is still running at Sat Apr 30 02:11:45 2022
child exited with status of 1
```

Full-Screen ScreenShot:



Conclusion:-

Thus, we have successfully implemented shell scripts to explore `wait()` and `waitpid()`.
