Summary of UNIX Fork function

what the function does

The UNIX Fork function is used to create new processes in UNIX-based operating system, and the function is accessible by using system call interfaces. After calling the function, it copies the calling process to create a new child process, and both processes are executing simultaneously.

The fork function **return two values**:

1).In the parent process, it returns the PID of process which is a positive integer or negative number if fork() fails to create a new process.

2).In the child process, it returns 0 if the child process is created successfully

Here is the code to initiate fork function:

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    printf("Before calling the fork(), the PID of current process is %d,"
                    "its parent process is %d", getpid(), getppid());
    pid_t pid = fork();

    if(pid > 0)
    {
        printf("This is the parent process."
                     "It created a child process with PID %d.\n", pid);
    } else if(pid == 0)
    {
        printf("This is the child process with PID %d and PPID %d.\n",
                        getpid(), getppid());
    } else
    {
        perror("fork failed");
        return 1;
    }
    return 0;
}
```

what activities expect in the OS

1).The process are expected to be in the running state in order to perform fork()

2).After executing the fork function, the OS create a new process(child process), and then both parent and child state are in the ready state for execution

3).Both process are expected in running statement during the execution, depends on the OS’ scheduling algorithm. Specifically, only one process can be in the running state per CPU core, if both process are scheduled to run on the same CPU core, it will run will run in a way that follows concurrency which involves rapidly switching between process.