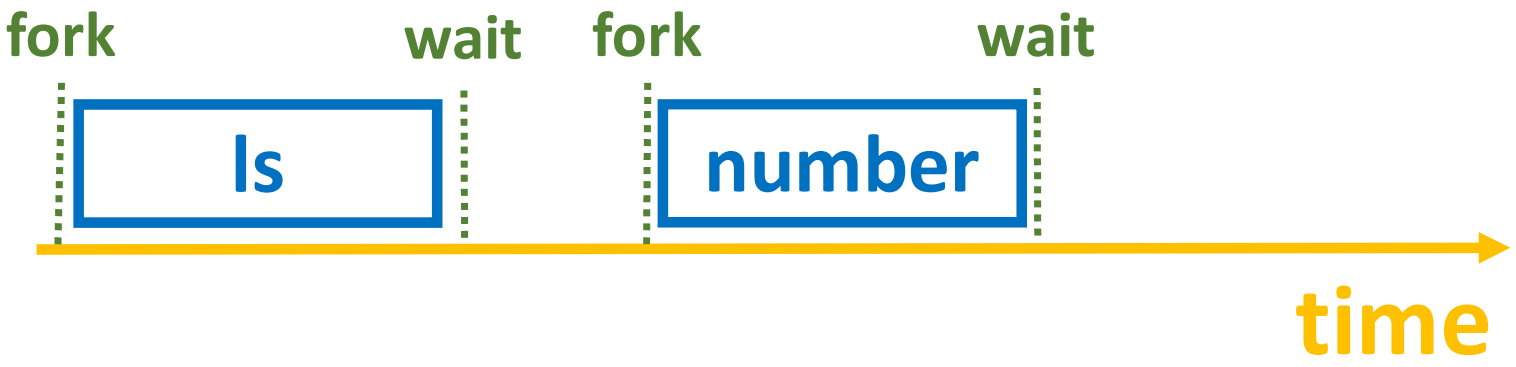
NP Project1 Supplementary

NP TA 詠嘉

Implementation 1: Wait for each child

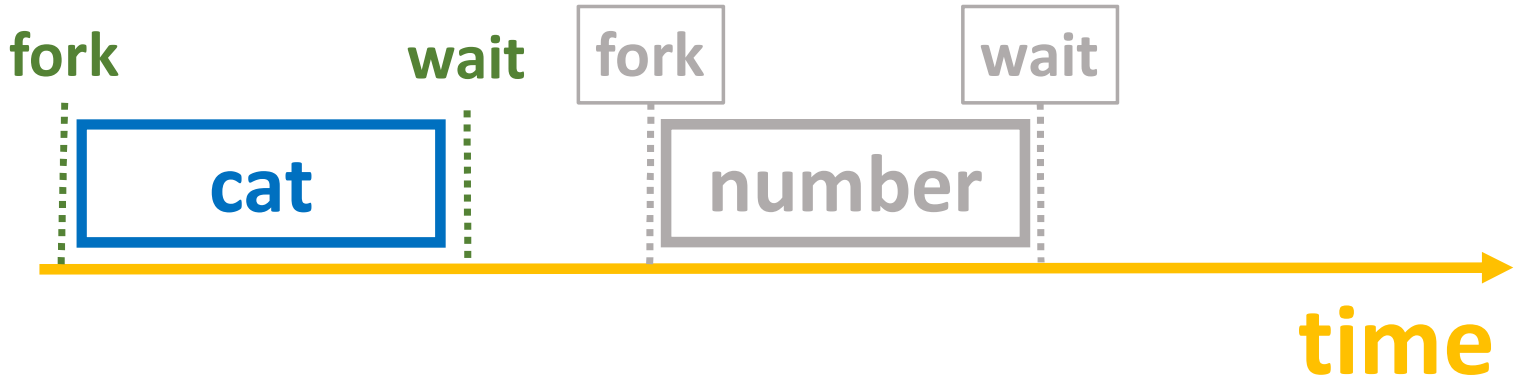
% ls | number



Problem 1: Unable to process large data

% cat largeFile.txt | number

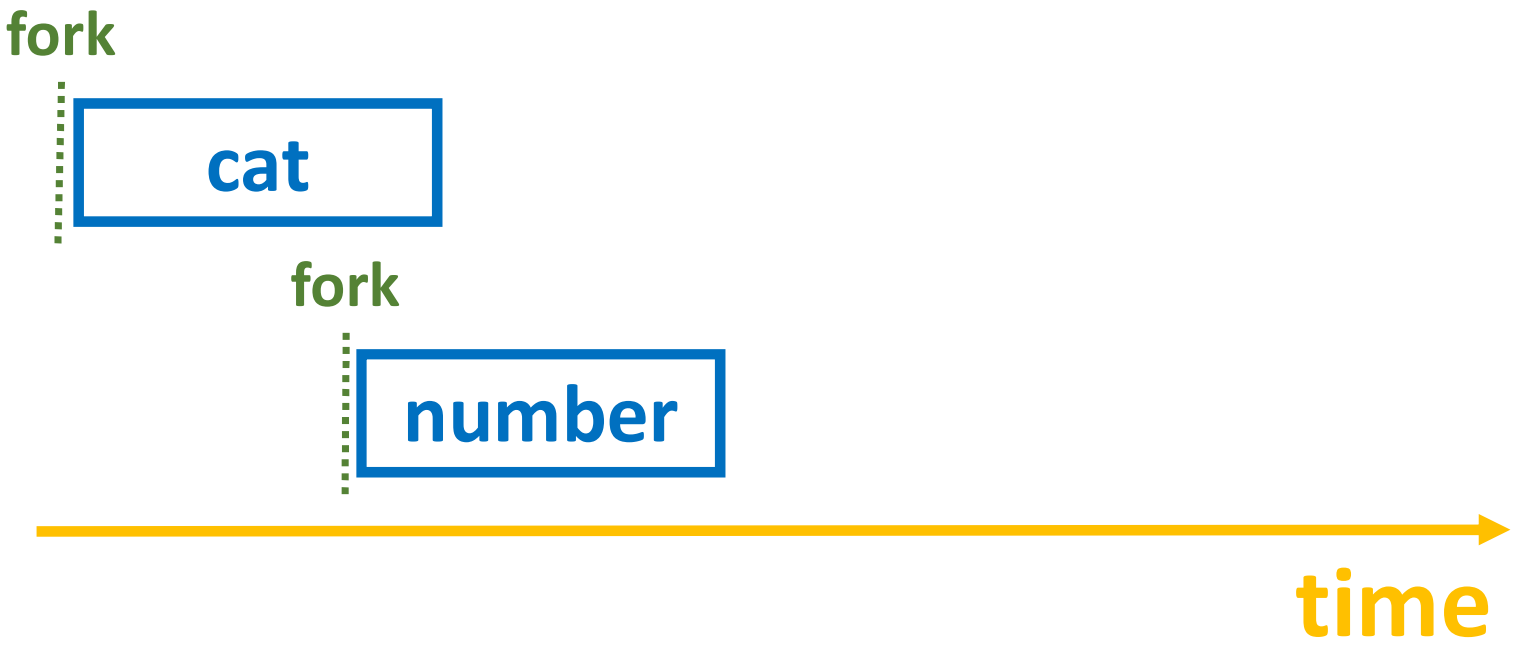
% cat largeFile.txt |1
% number



The process will hang forever !

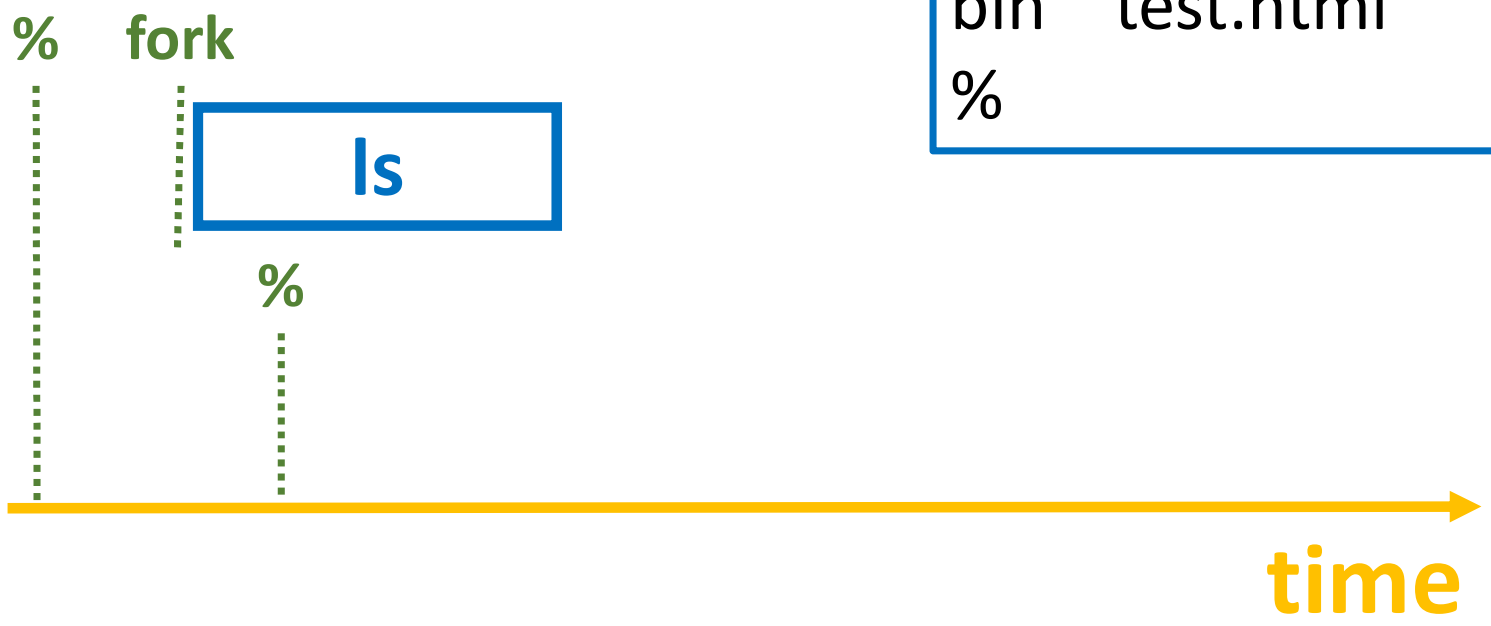
Implementation 2: Don’t wait for processes

% cat largeFile.txt | number



Problem 2 : % ordering

% ls
% bin test.html

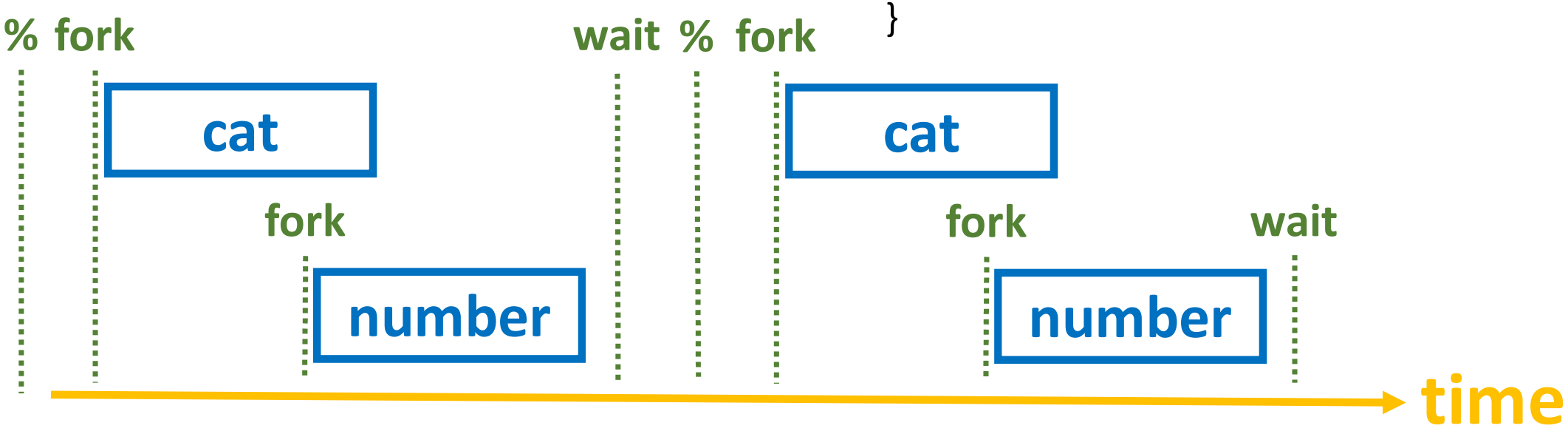


Correct:
% ls
bin test.html
%

Implementation 3 : Wait for a line if not pipeN

```
% cat largeFile.txt | number
% cat largeFile.txt |1
% number
```

```
forkAndExecAllCommandInCurrentLine();
If ( ! lineEndsWithPipeN) {
    for (pid in pidTableForCurrentLine) {
        int status;
        waitpid(pid, &status, 0);
    }
}
```

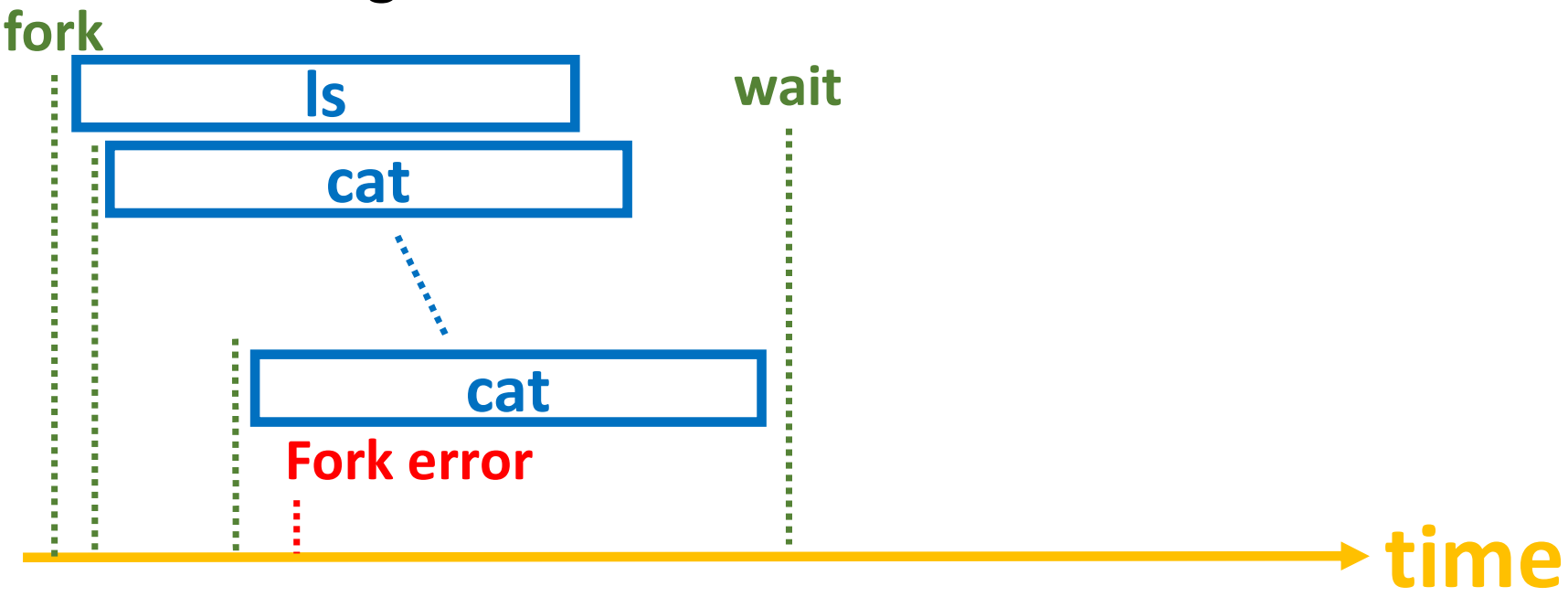


Problem 4 : Process limitation (1/2)

ls | cat | cat | cat | cat …… cat | cat

Process limit is **256** on NP server.

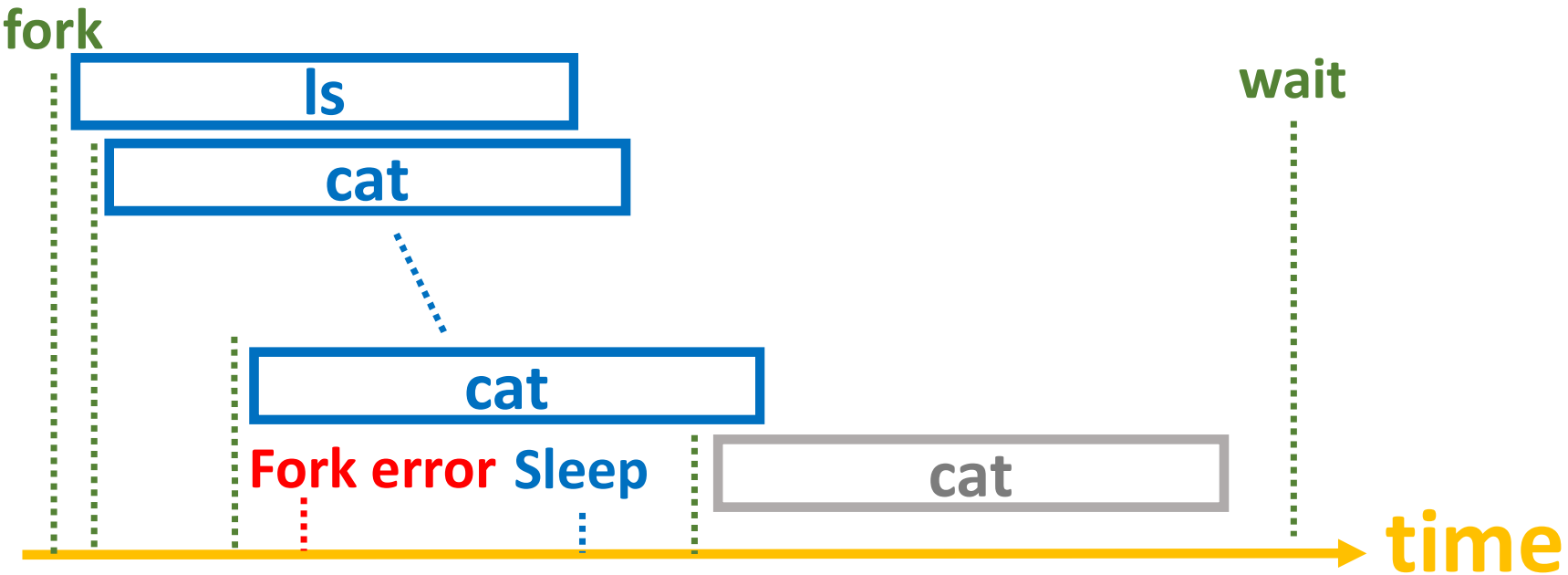
Fork error might occur around the 250th cat.



Implementation 5: If fork error, sleep a little

ls | cat | cat | cat | cat …… cat | cat

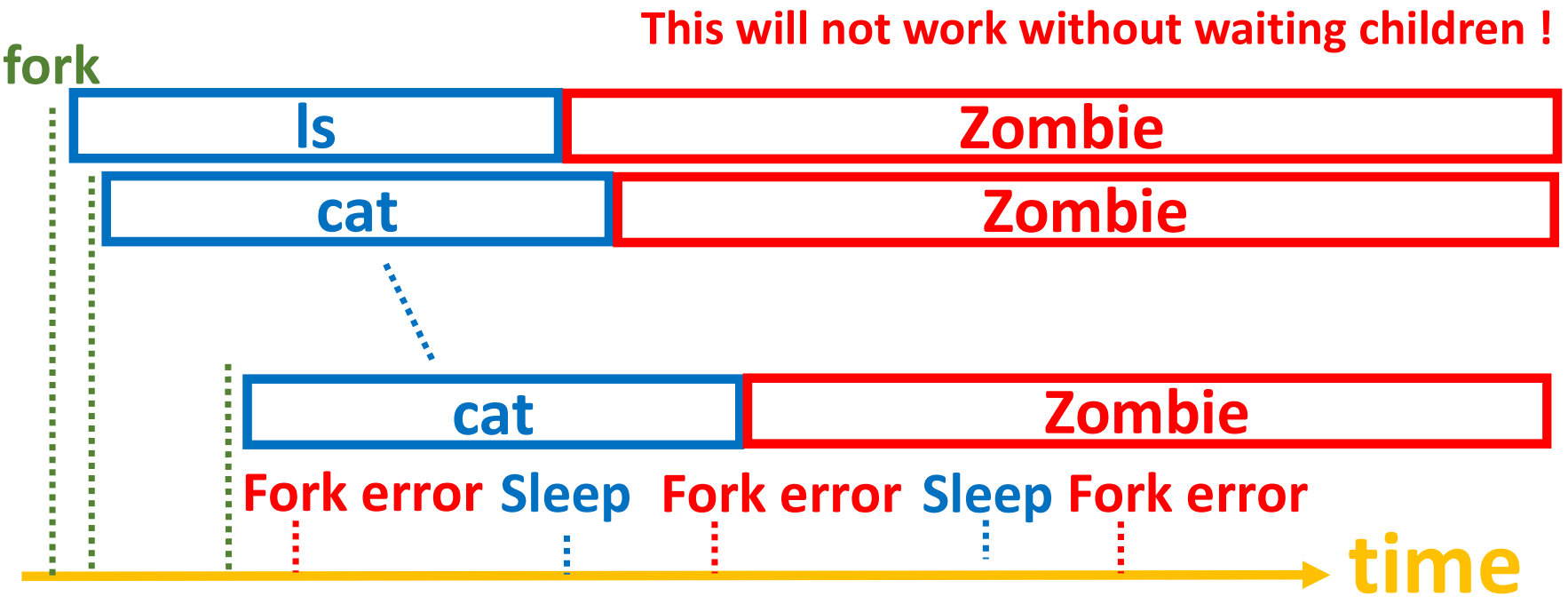
```
while ((pid = fork()) < 0) {
        usleep (1000);
}
```



Implementation 5: If fork error, sleep a little

ls | cat | cat | cat | cat …… cat | cat

```
while ((pid = fork()) < 0) {
        usleep (1000);
}
```



Implementation 6: Signal Handler

ls | cat | cat | cat | cat …… cat | cat

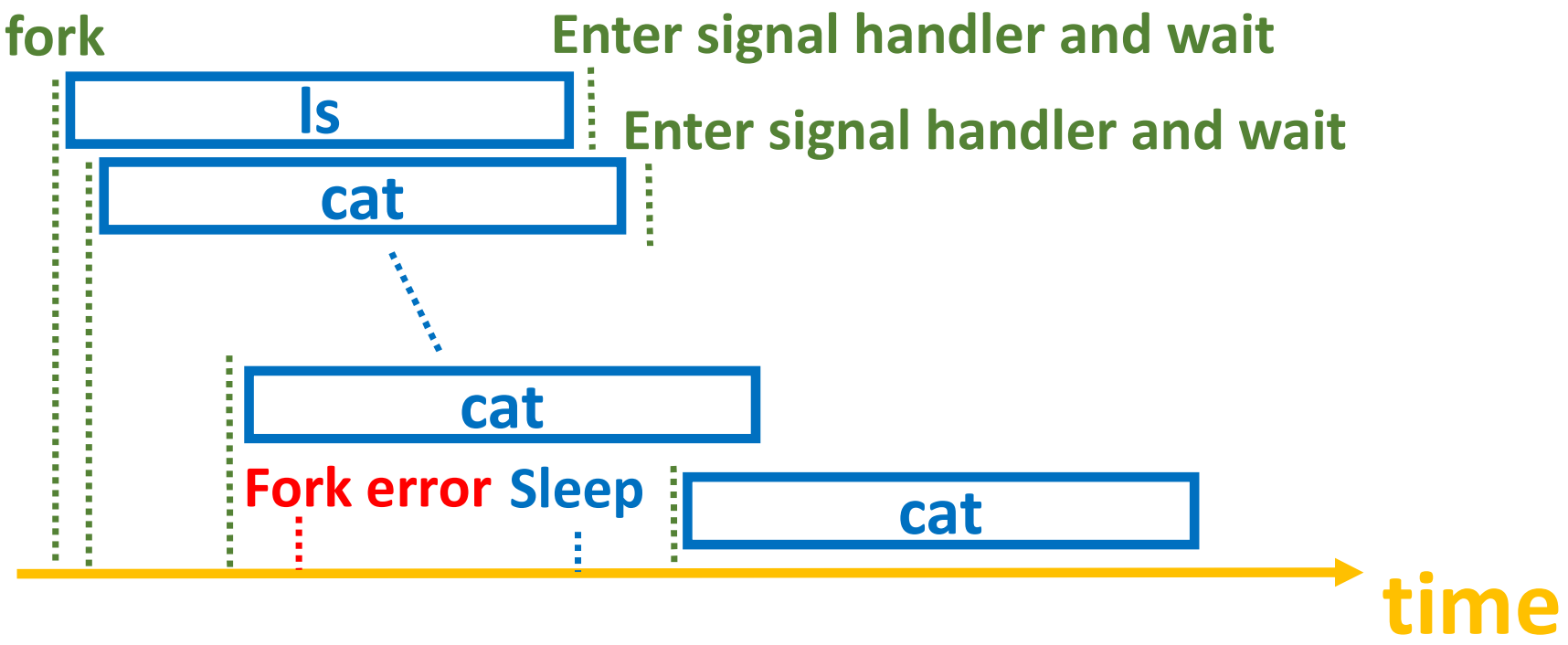
```
signal(SIGCHLD, childHandler);
……
while (fork() < 0) {
        usleep (1000);
}

……

If ( ! lineEndsWithPipeN) {
        for (pid in pidTableForCurrentLine) {
                int status;
                waitpid(pid, &status, 0);
        }
}
```

```
Void childHandler(int signo) {
        int status;
        while (waitpid(-1, &status, WNOHANG) > 0) {
                //do nothing
        }
}
```

-1: wait for any child process
WNOHANG: return immediately if no child has exited.

Implementation 6: Signal Handler

ls | cat | cat | cat | cat …… cat | cat



Implementation 6: Signal Handler

ls | cat | cat | cat | cat …… cat | cat

```
signal(SIGCHLD, childHandler);
……
while (fork() < 0) {
        usleep (1000);
}

……
```

```
If ( ! lineEndsWithPipeN) {
        for (pid in pidTableForCurrentLine) {
                int status;
                waitpid(pid, &status, 0);
        }
}
```

```
void childHandler(int signo) {
        int status;
        while (waitpid(-1, &status, WNOHANG) > 0) {
                //do nothing
        }
}
```

-1: wait for any child process
WNOHANG: return immediately if no child has exited.

Do we need both of this ?

Implementation 6: Signal Handler

ls | cat | cat | cat | cat …… cat | cat

```
signal(SIGCHLD, childHandler);
……
while (fork() < 0) {
        usleep (1000);
}

……
```

```
void childHandler(int signo) {
        int status;
        while (waitpid(-1, &status, WNOHANG) > 0) {
                //do nothing
        }
}
```

-1: wait for any child process
WNOHANG: return immediately if no child has exited.

```
If ( ! lineEndsWithPipeN) {
        for (pid in pidTableForCurrentLine) {
                int status;
                waitpid(pid, &status, 0);
        }
}
```

This is for solving % ordering, we need to keep this.

Implementation 6: Signal Handler

% cat largeFile.txt | number

largeFile.txt content…

% ls

bin test.html

%

```
void childHandler(int signo) {
    int status;
    while (waitpid(-1, &status, WNOHANG) > 0) {
        //do nothing
    }
}
```

```
If ( ! lineEndsWithPipeN) {
    for (pid in pidTableForCurrentLine) {
        int status;
        waitpid(pid, &status, 0);
    }
}
```