Child Monitoring

This homework will require you to write an application in C or C++ that spawns a child and then monitors its activity. The parent process must print the code corresponding to the action and, once the child terminates, the parent should print the time the child spent running on the CPU.

```
sam@sam-VirtualBox:~/Desktop$ ./a.out
Child terminated abnormally, without core dump.
child took: 7.690000 seconds
sam@sam-VirtualBox:~/Desktop$ ./a.out
Child stopped.
Child terminated abnormally, without core dump.
child took: 5.350000 seconds
```

Figure 1: Example output for the program. Your output should match the command output.

1. Your program **shall** spawn a single child that loops forever. You do not need to make any special signal handling in this child. It should simply spin in a `while` loop forever.

2. The parent process **shall** print code information each time it receives a signal about its child. You must print the following descriptions for the correct codes [1]:

Code	**Description**
CLD_CONTINUED	Child continued by SIGCONT
CLD_DUMPED	Child abnormal termination (core dump)
CLD_EXITED	Child exited
CLD_KILLED	Child abnormal termination (no core dump)
CLD_STOPPED	Child stopped
CLD_TRAPPED	Traced child stopped

3. Your program **shall** print the total CPU time consumed by the child process. Hopefully it is obvious that this should be done after the child terminates (otherwise it won't be correct).

4. The parent process **shall** efficiently wait for changes in the child process. The parent should not simply spin-loop. It should stop executing on the CPU if the child does not terminate, stop, continue, etc.

5. Your program **shall not** use `system` or the shell to perform any of its requirements.

6. Your program **shall** meet the basic requirements listed below:

   (a) Your program **shall** print reasonable error information.

[1] See Table 21-2 in your book.

(b) Your program **shall** terminate after encountering an unrecoverable error (after printing error information).

(c) You **shall** submit all source code and a makefile in a tar file.

   i. The makefile **shall** be able to build your source in a 32 bit Lubuntu VM.

   ii. The built program **shall** be able to run in a 32 bit x86 Lubuntu VM.