# ICSI Lab5 报告

计算机科学与技术
19307130296
孙若诗

## eval

功能为对用户输入的命令行解析并运行计算，对内置指令立即执行，否则 `fork` 一个子进程执行。基本框架已经在课本上给出，只需要完善进程阻塞和具体处理即可。

- 首先定义所需变量，初始化阻塞信号集合，将 `SIGCHLD` 加入 `mask_one` ， 将所有信号加入 `mask_all`
- 调用 `parseline` 解析 `cmdline`
- 调用 `builtin_cmd` 检查第一个命令行参数是否是一个内置指令。如果是，指令直接在 `builtin_cmd` 函数内被解释完成， `eval` 的工作结束
- 如果不是，需要创建一个子进程来执行请求的程序
  - 由于要在此进程 `addjob` ， 因此要先阻塞 `SIGCHLD` 防止竞争
  - `fork()` 返回 `0` 说明这是子进程
    - 使用 `setpgid` 创建一个新的进程组，方便后面使用 `kill` 发送信号
    - 解除进程阻塞
    - 使用 `execve` 执行 `cmdline` ， 无可执行文件则打印信息退出
  - 接下来要修改 `jobs` 相关的全局变量，因此先阻塞所有信号，再 `addjob`
  - 如果用户要求 `cmdline` 前台执行，则调用 `waitfg` 等候子进程结束
  - 如果用户要求后台执行，使用 `getjobpid` 函数获取任务信息，按 `rtest` 给出的格式打印
  - 处理结束，解除阻塞

## builtin_cmd

功能为判断输入指令是否为内置指令，是则立即解释并返回 `1` ， 否则返回 `0` ， 类似一个 `cmdline` 分类器。课本上同样给出了基本实现，只要再增加几种情况即可。

`shell lab` 中的 `shell` 内置的命令包括 `quit` 、 `jobs` 、 `bg` 、 `fg` 。

- 如果 `argv[0]` 为 `quit` ， 直接退出
- 如果 `argv[0]` 为 `jobs` ， 按要求调用 `listjobs` 打印所有 `jobs` 信息，返回 `1`
- 如果 `argv[0]` 为 `bg` 或 `fg` ， 调用 `do_bgfg` 处理，返回 `1`
- 如果 `argv[0]` 不满足上述情况，返回 `0`

# do_bgfg

功能为执行 bg 和 fg 指令。

- 首先判断 argv[1] 是否为 NULL ， 是则打印错误信息并返回
- 按照 shlab.pdf 的说明，用第一位是否为 % 判断 id 属于作业还是进程，用 atoi 由字符数组类型转换为 int ， 若 id 为 0 则打印错误信息并返回
- 接下来要查询 jobs 相关信息，因此先阻塞信号，函数返回前消除阻塞
- 使用 getjobjid 或 getjobpid 获取对应作业，获取失败则打印错误信息并返回
- 若 cmd 指令为 fg ， 将 job 状态设置为 FG ， 用 kill 向该进程组发送 SIGCONT 信号，等待该进程执行结束后返回
- 若 cmd 指令为 bg ， 将 job 状态设置为 BG ， 打印相关信息，用 kill 向该进程组发送 SIGCONT 信号

# waitfg

功能为等待前台程序运行结束。参数为 pid ， 即当前监控的进程。

- 由于 jobs 为全局变量，查询之前先阻塞所有信号，查询后再解除。
- 调用 fgpid ， 其功能为返回目前前台运行的进程id，如果没有前台进程返回 0 。 waitfg 的目标即为 pid 等于 fgpid 时保持休眠。
- 休眠用 sigsupend 实现，创建一个空的阻塞信号集合作为参数，在捕捉到信号之前将进程挂起

# sigchld_handler

功能为响应 SIGCHLD 信号。

- 首先按照课本上的安全信号处理原则，将 errno 记录，退出函数之前还原
- 调用 waitpid 函数，使用 WNOHANG | WUNTRACED 参数，立即返回，如果等待集合中的子进程都没有停止或终止，返回 0 ； 如果有一个被停止或终止，返回该子进程的 pid
- 如果 pid 大于 0 ， 说明有停止或终止的子进程，可以从 statusp 获取相关信息
- 调用 WIFSTOPPED ， 如果子进程被停止返回 1 ， 用 getjobpid 获取作业信息、 WSTOPSIG 获取引起子进程停止的信息编号，并打印
- 否则子进程是终止的
  - 调用 WIFSIGNALED ， 如果子进程是因为一个未被捕获的信号终止的返回 1 ， 用 WTERMSIG 获取导致子进程终止的信息编号并打印
  - 从作业列表中删除该子进程
- 查询后阻塞信号，以备 deletejob 修改全局变量，处理结束后解除阻塞

# sigint_handler

功能为响应 SIGINT(Ctrl+C) 。

- 记录和还原 errno
- 使用 fgpid 查询当前在前台执行的进程。由于 fgpid 涉及到全局变量，照例阻塞所有信号，查询后再解除
- 如果当前有前台进程，使用 kill 对该进程所在的进程组发送 SIGINT 信号，要求进程终止

# sigtstp_handler

功能为响应 SIGTSTP(Ctrl+Z) ，除 kill 发送的信息改为 SIGTSTP 之外均和 sigint_handler 相同，意为使进程停止直到下一个 SIGCONT 。