简述如何在 Linux Bash 脚本中监控后台进程

张文帅（中国科学技术大学超算中心）

September 26, 2018

Abstract

利用后台运行程序的方法，可大大增加作业的自由度。但是，如果后台进程无法被有效监控与终止，会造成子进程或子子进程遗留，系统资源负载异常，甚至某些后台程序持续运行会导致结果不符合预期，出现错误。所以一旦开始使用后台运行的方式，必须要保证父子继承的一系列后台进程完全可控。本文档简述如何递归的抓取所有子进程、孙进程 ^_^，并在预定条件时终止。

1 后台进程

简单了解下后台进程相关的命令:

- 命令使用 & 来结束，使其后台运行
- jobs 查看当前进程下启动的后台进程
- $$ 记录当前进程的 PID
- $! 记录上一个后台进程的 PID
- pgrep -P PID 检查进程 PID 的子进程

```
Bash4.2:$ sleep 600 &
[1] 16895
Bash4.2:$ jobs -l
[1]+ 16895 Running sleep 600 &
Bash4.2:$ sleep 500 &
[2] 16937
Bash4.2:$ jobs -l
[1]- 16895 Running sleep 600 &
[2]+ 16937 Running sleep 500 &
Bash4.2:$ echo $$
24020
Bash4.2:$ pgrep -P 24020
16895
16937
Bash4.2:$ sleep 400 &
[3] 21212
Bash4.2:$ echo $!
21212
```

2 几个有用的函数

- 获取递归获取某 PID 的子进程

```
function getchild_recurse () {
    local PID="$1"
    local printself="${2:-False}"
    local Pchild_list
    local child
    printf "\n "
    [ "$printself" == "True" ] && printf " $PID"
    if Pchild_list=( $(pgrep -P $PID) ); then
        printf " ${Pchild_list[*]}"
        for child in ${Pchild_list[*]}; do
            getchild_recurse "$child"
        done
    fi
    return 0
}
```

测试: 第二个测试中的第二个参数 "True" 代表也输出打印输入的 PID 进程号

```
Bash4.2:$ getchild_recurse 24020
  27594 27596 27600 27608

  27751 28236

Bash4.2:$ getchild_recurse 24020 "True"
  24020 27594 27596 27600 27608

  27751 28236
```

- 递归终止进程，第二个参数为“True”时表示亦终止输入的 PID 进程，否则只终止其所有递归子进程

```
function killchild_recurse () {
    local PID="$1"
    local killself="${2:-False}"
    local Pchild_list
    local child
    if Pchild_list="$(pgrep -P $PID)"; then
        for child in $Pchild_list; do
            printf "\n $PID haschild: $child "
            killchild_recurse "$child" "True"
        done
    fi
    if [ "$killself" == "True" ]; then
        printf "\n kill -9 $PID "
        kill -9 "$PID"
```

```
    fi
    return 0
}
```

- 检查进程列表，如存在任何一个进程，则成功返回，否则错误返回。

```
function hasPIDs () {
    local PID
    for PID in $1
    do
        if ps -P $PID > /dev/null ; then
            return 0
        fi
    done
    return 1
}
```

3 示例代码，监控某后台进程

当运行时间超过设定值时，终止所有后台进程。可按需要修改示例代码，添加监控条件。

```
function bgrun_cmd () {
    local runcmd="$1"
    local timelimit=$2
    local runcmd_childs
    local Tinterval=15
    local ntime=0
    local PID
    local iPID
    local killjob="False"

    echo -e "\n run: $runcmd "
    $runcmd &
    PID=$!
    runcmd_childs=( $(getchild_recurse "$PID") )
    printf "\n PID: $PID (${runcmd_childs[*]}) running bg @ $$/$(hostname) \n"

    while hasPIDs "${runcmd_childs[*]}"
    do
        sleep $Tinterval
        ntime=$(($ntime+1))

        if [ `expr $ntime \* $Tinterval` -ge "$timelimit" ]; then
            printf "\n PID: $PID timeout beyond ${timelimit}s, kill & exit \n"
            killjob="True"
        fi
```

```
        # other condition or monitor ... for example:
        if [ `grep "LOOP:" "./OUTCAR" | wc -l` -ge 10 ]; then
            printf "\n Number of LOOP >= 10, kill & exit \n"
            killjob="True"
        fi

        if [ "$killjob" == "True" ]; then
            printf "\n Kill PID: $PID (${runcmd_childs[*]}) \n"
            killchild_recurse "$PID" "True"
            sleep 15
            while hasPIDs "${runcmd_childs[*]}"
            do
                sleep 15
                for iPID in ${runcmd_childs[*]}; do
                    kill -9 $iPID
                done
            done
            printf "\n PID: $PID was killed \n"
            break
        fi
    done
    echo " PID: $PID was exited successfully"
    return 0
}
```