链滴

golang command

作者：jssyjam

原文链接：https://ld246.com/article/1568718412370

来源网站：链滴

golang中会经常遇到要 fork 子进程的需求。go 标准库为我们封装了 os/exec标准包,当我们要运行部命令时应该优先使用这个库。

执行 command(超时处理)

这里我简单结合context 和 Cmd 模块写一个通用的执行 command 方法。代码如下:

```
package main

import (
    "context"
    "os/exec"
    "syscall"
)

func RunCmd(ctx context.Context, cmd *exec.Cmd) error {
    cmd.SysProcAttr = &syscall.SysProcAttr{
        Setpgid: true,
    }

    if err := cmd.Start(); err != nil {
        return err
    }

    errCh := make(chan error, 1)
    go func() {
        errCh <- cmd.Wait()
    }()

    done := ctx.Done()
    for {
```

```
        select {
        case <-done:
            done = nil
            pid := cmd.Process.Pid
            if err := syscall.Kill(-1*pid, syscall.SIGKILL); err != nil {
                return err
            }
        case err := <-errCh:
            if done == nil {
                return ctx.Err()
            } else {
                return err
            }
        }
    }
}
```

说明:

- 可以通过 context 控制命令执行, 调用方可以调用 cancel 或者设置超时控制命令执行生命周期
- 如果进程执行失败, 应当 kill 整个进程组, 防止该进程 fork 的子进程逃逸

执行用户设置

如果需要指定执行用户可以添加SysProcAttr的Credential属性。具体代码如下:

```
func withSysProcAttr(cmd *exec.Cmd, userName string) error {
  // 检测用户是否存在
    user, err := user.Lookup(userName)
    if err != nil {
        return errors.Wrapf(err, "invalid user %s", userName)
    }

    // set process attr
  // 获取用户 id
    uid, err := strconv.ParseUint(user.Uid, 10, 32)
    if err != nil {
        return err
    }
  // 获取用户组 id
    gid, err := strconv.ParseUint(user.Gid, 10, 32)
    if err != nil {
        return err
    }

    attr := getSysProcAttr()
    //设置进程执行用户
    attr.Credential = &syscall.Credential{
        Uid:         uint32(uid),
        Gid:         uint32(gid),
        NoSetGroups: true,
    }
```

```
    cmd.SysProcAttr = attr
    return nil
}
// 设置进程组属性
func getSysProcAttr() *syscall.SysProcAttr {
    return &syscall.SysProcAttr{
        Setpgid: true,
    }
}
```

Tips:设置进程执行用户前首先检查用户是否存在