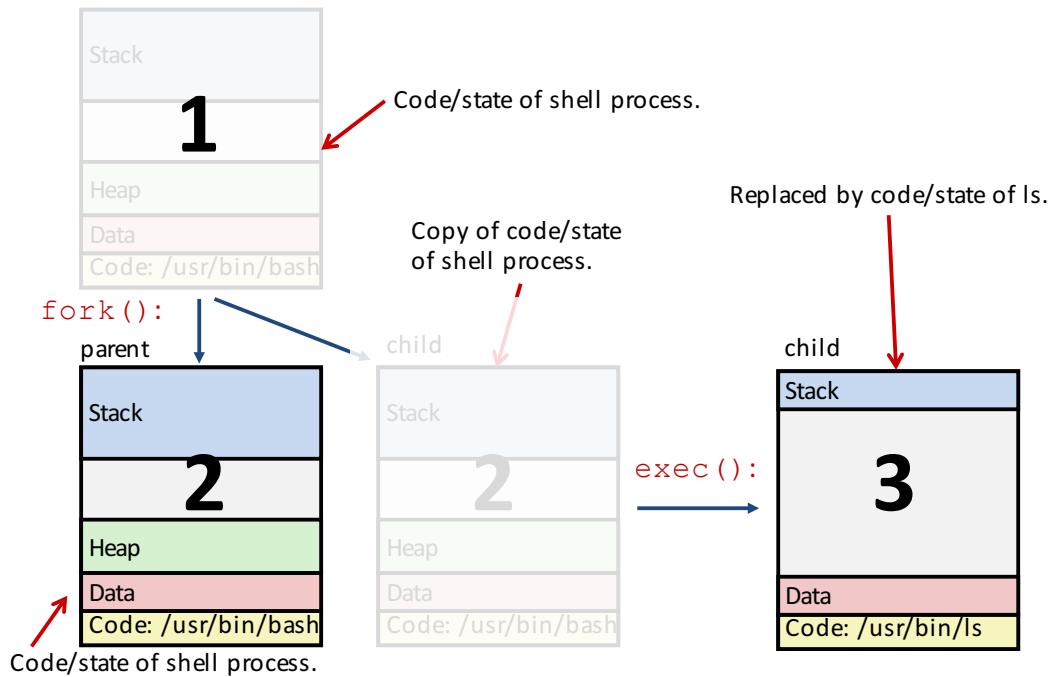


Fork/Exec



Reaping: waitpid / wait

```
pid_t waitpid(pid_t pid, int* stat, int ops)
```

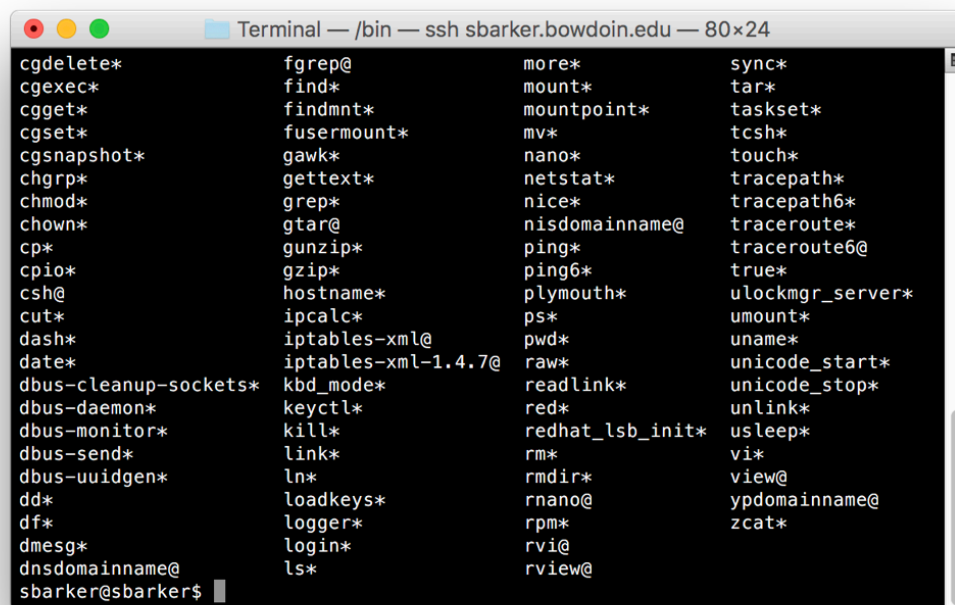


System Call Error Handling

Always check return values!

```
if ((pid = fork()) < 0) {  
    fprintf(stderr, "fork error: %s\n", strerror(errno));  
    exit(0);  
}
```

Shell



Terminal — /bin — ssh sbarker.bowdoin.edu — 80x24

```
cgdelete*      fgrep@      more*        sync*  
cgexec*        find*       mount*       tar*  
cgget*         findmnt*    mountpoint* taskset*  
cgset*         fusermount* mv*          tcsh*  
cgsnapshot*    gawk*       nano*        touch*  
chgrp*         gettext*    netstat*     tracepath*  
chmod*         grep*       nice*        tracepath6*  
chown*         gtar@       nisdomainname@ traceroute*  
cp*            gunzip*     ping*        traceroute6@  
cpio*          gzip*       ping6*       true*  
csh@           hostname*   plymouth*    ulockmgr_server*  
cut*           ipcalc*     ps*          umount*  
dash*          iptables-xml@ pwd*         uname*  
date*          iptables-xml-1.4.7@ raw*         unicode_start*  
dbus-cleanup-sockets* kbd_mode*  readlink*   unicode_stop*  
dbus-daemon*   keyctl*     red*         unlink*  
dbus-monitor*  kill*       redhat_lsb_init* usleep*  
dbus-send*     link*       rm*          vi*  
dbus-uuidgen*  ln*         rmdir*       view@  
dd*            loadkeys*   rnano@       ypdomainname@  
df*            logger*     rpm*         zcat*  
dmesg*         login*      rvi@  
dnsdomainname@ ls*         rview@  
sbarker@sbarker$
```

Shell Design

```
while (true) {  
    Print command prompt.  
    Read command line from user.  
    Parse command line.  
    If command is built-in, do it.  
    Else fork process to execute command.  
        in child:  
            Execute requested command with execv.  
                (never returns)  
        in parent:  
            Wait for child to complete.  
}
```