# How BASH starts up

# Interactive Login Shell*

After *login*:

```
if [ -f /etc/profile ] ; then source /etc/profile; fi

if [ -r ~/.bash_profile ] ; then
    source ~/.bash_profile;
elif [ -r ~/.bash_login ] ; then
    source ~/.bash_login;
elif [ -r ~/.profile ] ; then
    source ~/.profile;
fi
```

After *exit*:

```
if [ -f ~/.bash_logout ] ; then source ~/.bash_logout; fi
```

*Interactive login shell is started after successful login
via /bin/login
-OR-
As interactive non-login shell with --login option

# Interactive Non-Login Shell

```
if [ -f ~/.bashrc ] ; then . ~/.bashrc; fi
```

# Non-Interactive Shell

```
if [ -f $BASH_ENV ] ; then . $BASH_ENV; fi
```

# Invoked with name sh

Tries to mimic the startup behavior of historical versions of sh as closely as possible, while conforming to the POSIX standard as well.

As ILS or as (n-)IS with *--login*:

```
if [ -f /etc/profile ] ; then source /etc/profile; fi
if [ -f ~/.profile ] ; then source ~/.profile; fi
```

As IS:

```
if [ -f $ENV ] ; then source $ENV; fi
```

As n-IS:

tries to read nothing

# Invoked in POSIX mode

When Bash is started in POSIX mode, as with the --posix command line option, it follows the POSIX standard for startup files.

```
if [ -f $ENV ] ; then source $ENV; fi
```

# Invoked by remote shell daemon (rshd/sshd)

Bash attempts to determine when it is being run with its standard input connected to a a network connection, as if by the remote shell daemon, usually rshd, or the secure shell daemon sshd.

```
if [ -f ~/.bashrc ] ; then source ~/.bashrc; fi
```

*It will not do that if invoked as *sh*.