Linux Bashrc 配置

这个`.bashrc`文件配置 Linux 中的 Bash shell 环境。它自定义提示符、设置别名、管理代理设置，并与 Git 等工具集成。以下是关键配置的细分：

1. 基本设置：

- `HISTCONTROL=ignoreboth`: 忽略历史记录中重复的命令和以空格开头的命令。
- `shopt -s histappend`: 将新的历史记录条目追加到历史记录文件。
- `HISTSIZE=1000`: 将内存中保留的历史记录条目数设置为 1000。
- `HISTFILESIZE=2000`: 将历史记录文件的最大大小设置为 2000。
- `shopt -s checkwinsize`: 更新终端窗口大小。

2. 彩色提示符：

- 如果终端支持，则配置彩色命令提示符。

3. 窗口标题：

- 将终端窗口标题设置为显示当前用户、主机和工作目录。

4. 目录颜色：

- 如果可用，则为 `ls` 命令启用彩色输出。

5. 别名：

- `alias ll='ls -alF'`: 列出所有文件，并显示详细信息。
- `alias la='ls -A'`: 列出所有文件，包括隐藏文件。
- `alias l='ls -CF'`: 列出文件，以列的形式显示。
- `alias alert='notify-send ...'`: 命令结束后发送桌面通知。

6. Bash 别名文件：

- 包含一个用于自定义别名的单独文件（`~/.bash_aliases`）。

7. Bash 自动补全：

- 如果可用，则启用 bash 自动补全。

8. 路径配置：

- `export PATH=...`: 将各种目录添加到 `PATH` 环境变量中，包括 CUDA、Ruby gems、本地二进制文件和系统二进制文件的目录。

9. 代理管理：

- `export GLOBAL_PROXY='127.0.0.1:7890'`: 定义代理服务器地址的变量。
- `function start_proxy { ... }`: 设置 `HTTP_PROXY`、`HTTPS_PROXY`、`http_proxy`、`https_proxy` 和 `ALL_PROXY` 环境变量以使用指定的代理。
- `function start_proxy_without_prefix { ... }`: 与 `start_proxy` 类似，但是设置代理变量时不带 `http://` 前缀。
- `function stop_proxy { ... }`: 取消设置代理变量，有效地禁用代理。
- `export NO_PROXY="localhost,127.0.0.1,.example.com,::1"`: 指定应绕过代理的主机。

10. Git 代理：

- `function start_git_proxy { ... }`: 配置git以使用全局代理进行HTTP和HTTPS连接。
- `function stop_git_proxy { ... }`: 取消设置git代理设置。

11. 默认代理：

- `start_proxy`: 默认启动代理。
- `start_git_proxy`: 默认启动git代理。

12. Python 别名：

- `alias python=python3`: 将 `python` 设置为使用 `python3`。
- `alias pip=pip3`: 将 `pip` 设置为使用 `pip3`。

13. Git 消息 AI 别名：

- `function gpa { ... }`: 创建一个别名 `gpa` 来运行 python 脚本 `gitmessageai.py`，使用 mistral API，并允许拉取和推送。
- `function gca { ... }`: 创建一个别名 `gca` 来运行相同的脚本，但不推送更改。
- `function gm { ... }`: 创建一个别名 `gm` 来运行相同的脚本，并且只打印提交消息。

14. 使用拉取和变基的 Git 推送：

- `function gpp { ... }`: 尝试推送更改，如果失败，则尝试使用变基拉取，然后再次推送。

15. 预执行代理检查：

- `preexec() { ... }`: 此函数在每个命令之前执行。它检查命令是否在网络相关命令列表中。如果是，并且设置了任何代理变量，则显示代理设置。
- `local network_commands=( ... )`: 此数组列出了被认为是网络相关的命令。
- `display_proxy() { ... }`: 此函数显示当前的代理设置。

16. 检查代理函数：

- `function checkproxy { ... }`: 显示当前的 HTTP 和 HTTPS 代理设置，以及 Git 代理设置。

```
case $- in
    *i*) ;;
      *) return;;
esac

HISTCONTROL=ignoreboth
shopt -s histappend
HISTSIZE=1000
HISTFILESIZE=2000
shopt -s checkwinsize

[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"

if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
    color_prompt=yes
    else
    color_prompt=
    fi
fi

if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

case "$TERM" in
```

```
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac

if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|s

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi



export PATH="/usr/local/cuda-12.2/bin:/home/lzw/.local/share/gem/ruby/3.0.0/bin:/home/lzw/.local/bin:/usr/loca

export GLOBAL_PROXY='127.0.0.1:7890'
```

```
function start_proxy {
    export HTTP_PROXY="http://$GLOBAL_PROXY"
    export HTTPS_PROXY="http://$GLOBAL_PROXY"
    export http_proxy="http://$GLOBAL_PROXY"
    export https_proxy="http://$GLOBAL_PROXY"
    export HTTP_PROXY_REQUEST_FULLURI=false
    export HTTPS_PROXY_REQUEST_FULLURI=false
    export ALL_PROXY=$http_proxy
}

function start_proxy_without_prefix {
    export http_proxy=$GLOBAL_PROXY
        export HTTP_PROXY=$GLOBAL_PROXY
        export https_proxy=$GLOBAL_PROXY
    export HTTPS_PROXY=$GLOBAL_PROXY
    export HTTP_PROXY_REQUEST_FULLURI=false
    export HTTPS_PROXY_REQUEST_FULLURI=false
        export ALL_PROXY=$http_proxy
}

function stop_proxy {
    export http_proxy=
        export HTTP_PROXY=
        export https_proxy=
    export HTTPS_PROXY=
    export HTTP_PROXY_REQUEST_FULLURI=true
    export HTTPS_PROXY_REQUEST_FULLURI=true
        export ALL_PROXY=
}

export NO_PROXY="localhost,127.0.0.1,.example.com,::1"


function start_git_proxy {
  git config --global http.proxy $GLOBAL_PROXY
  git config --global https.proxy $GLOBAL_PROXY
}

function stop_git_proxy {
  git config --global --unset http.proxy
```

```
  git config --global --unset https.proxy
}

start_proxy
start_git_proxy

alias python=python3
alias pip=pip3

function gpa {
  python ~/bin/gitmessageai.py --api mistral --allow-pull-push
}

function gca {
  python ~/bin/gitmessageai.py --no-push
}

function gm {
  python ~/bin/gitmessageai.py --only-message
}

function gpp {
  git push || {
    echo "Push failed, attempting pull and merge"
    git pull --rebase || {
      echo "Pull failed, please resolve conflicts manually"
      return 1
    }
    git push || {
      echo "Push failed after pull, please resolve conflicts manually"
      return 1
    }
  }
}

preexec() {
    local network_commands=(
        "gpa"
        "git"
        "ssh"
```

```
    "scp"
    "sftp"
    "rsync"
    "curl"
    "wget"
    "apt"
    "yum"
    "dnf"
    "npm"
    "yarn"
    "pip"
    "pip3"
    "gem"
    "cargo"
    "docker"
    "kubectl"
    "ping"
    "traceroute"
    "netstat"
    "ss"
    "ip"
    "ifconfig"
    "dig"
    "nslookup"
    "nmap"
    "telnet"
    "ftp"
    "nc"
    "tcpdump"
    "adb"
    "bundle"
    "brew"
    "cpanm"
    "bundle exec jekyll"
    "make"
    "python"
    "glcoud"
)

local cmd
```

```
    cmd=$(echo "$1" | awk '{print $1}')

    display_proxy() {
        echo -e "  **Proxy Settings Detected:**"

        [ -n "$HTTP_PROXY" ] && echo "    - HTTP_PROXY: $HTTP_PROXY"
        [ -n "$HTTPS_PROXY" ] && echo "    - HTTPS_PROXY: $HTTPS_PROXY"

        echo ""
    }

    for network_cmd in "${network_commands[@]}"; do
        if [[ "$1" == "$network_cmd"* ]]; then
            if [ -n "$HTTP_PROXY" ] || [ -n "$http_proxy" ] || \
               [ -n "$HTTPS_PROXY" ] || [ -n "$https_proxy" ] || \
               [ -n "$ALL_PROXY" ] || [ -n "$all_proxy" ]; then

                display_proxy
            fi
            break
        fi
    done
}

function checkproxy {
  echo "HTTP_PROXY: $HTTP_PROXY"
  echo "HTTPS_PROXY: $HTTPS_PROXY"
  echo "Git HTTP Proxy:"
  git config --get http.proxy
  echo "Git HTTPS Proxy:"
  git config --get https.proxy
}
```