Linux Bashrc 設定

この `bashrc` ファイルは、Linux における Bash シェルの環境を設定します。プロンプトのカスタマイズ、エイリアスの設定、プロキシ設定の管理、Git などのツールとの統合を行います。以下に主要な設定の内訳を示します。

1. 基本設定:

- `HISTCONTROL=ignoreboth`: 履歴から重複したコマンドとスペースで始まるコマンドを無視します。
- `shopt -s histappend`: 新しい履歴エントリを履歴ファイルに追加します。
- `HISTSIZE=1000`: メモリに保持する履歴エントリの数を設定します。
- `HISTFILESIZE=2000`: 履歴ファイルの最大サイズを設定します。
- `shopt -s checkwinsize`: ターミナルウィンドウのサイズを更新します。

2. カラプロンプト:

- ターミナルがサポートしている場合、カラーコマンドプロンプトを設定します。

3. ウィンドウタイトル:

- ターミナルウィンドウのタイトルに、現在のユーザー、ホスト、作業ディレクトリを表示するように設定します。

4. ディレクトリの色:

- `dircolors` が利用可能な場合、`ls` コマンドの出力に色を有効にします。

5. エイリアス:

- `alias ll='ls -alF'`: 詳細な情報を含むすべてのファイルをリストします。
- `alias la='ls -A'`: 隠しファイルを含むすべてのファイルをリストします。
- `alias l='ls -CF'`: ファイルを列形式でリストします。
- `alias alert='notify-send ...'`: コマンドの実行後にデスクトップ通知を送信します。

6. Bash エイリアスファイル:

- カスタムエイリアス用の別々のファイル (`~/.bash_aliases`) を含めます。

7. Bash 補完:

- 利用可能な場合、Bash 補完を有効にします。

8. パス設定:

- `export PATH=...`: CUDA、Ruby gem、ローカルバイナリ、システムバイナリなど、様々なディレクトリを`PATH`環境変数に追加します。

9. プロキシ管理:

- `export GLOBAL_PROXY='127.0.0.1:7890'`: プロキシサーバーのアドレスを変数に定義します。
- `function start_proxy { ... }`: 指定されたプロキシを使用するために、`HTTP_PROXY`、`HTTPS_PROXY`、`http_proxy`、`https_proxy`、`ALL_PROXY`環境変数を設定します。
- `function start_proxy_without_prefix { ... }`: `start_proxy`と同様ですが、`http://`プレフィックスなしでプロキシ変数を設定します。
- `function stop_proxy { ... }`: プロキシ変数をアンセットし、プロキシを無効にします。
- `export NO_PROXY="localhost,127.0.0.1,.example.com,::1"`: プロキシをバイパスする必要があるホストを指定します。

10. Git プロキシ:

- `function start_git_proxy { ... }`: HTTPおよびHTTPS接続にグローバルプロキシを使用するようにGitを設定します。
- `function stop_git_proxy { ... }`: Gitプロキシ設定をアンセットします。

11. デフォルトプロキシ:

- `start_proxy`: デフォルトでプロキシを開始します。
- `start_git_proxy`: デフォルトでGitプロキシを開始します。

12. Python エイリアス:

- `alias python=python3`: `python`を `python3`を使用するように設定します。
- `alias pip=pip3`: `pip`を `pip3`を使用するように設定します。

13. Git メッセージ AI エイリアス:

- `function gpa { ... }`: Mistral API を使用して `gitmessageai.py` Python スクリプトを実行し、プルとプッシュを許可するエイリアス `gpa` を作成します。
- `function gca { ... }`: 変更をプッシュせずに同じスクリプトを実行するエイリアス `gca` を作成します。
- `function gm { ... }`: 同じスクリプトを実行し、コミットメッセージのみを出力するエイリアス `gm` を作成します。

14. プルとリベースによる Git プッシュ:

- `function gpp { ... }`: 変更をプッシュしようとします。失敗した場合は、リベースしてプルし、再度プッシュしようとします。

15. 実行前のプロキシチェック:

- `preexec() { ... }`: この関数は、すべてのコマンドの前に実行されます。ネットワーク依存コマンドのリストにコマンドが含まれているかどうかをチェックします。含まれており、プロキシ変数が設定されている場合は、プロキシ設定を表示します。
- `local network_commands=( ... )`: この配列は、ネットワーク依存と見なされるコマンドをリストします。
- `display_proxy() { ... }`: この関数は、現在のプロキシ設定を表示します。

16. プロキシチェック関数:

- `function checkproxy { ... }`: 現在の HTTP および HTTPS プロキシ設定、ならびに Git プロキシ設定を表示します。

```
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
}
```