# プロキシチェックの実装

Windows で Git-Bash と PowerShell を使用してプロキシチェックを実装するには、各シェルでネットワーク関連のコマンドが実行される際にプロキシ設定を検出し、設定されている場合に表示するメカニズムを設定する必要があります。以下に、Git-Bash と PowerShell の手順とコードが示されています。「ターミナル」とは、これらのシェルをホストする Windows Terminal を指していると考えられますので、Git-Bash と PowerShell の実装に焦点を当てます。

---

### Git-Bash の場合

Git-Bash は Windows 上の Bash エミュレーションであり、`DEBUG` トラップを使用して各コマンド実行前に関数を実行することができます。目標は、コマンドがネットワーク関連であるかどうかを確認し、プロキシ設定が設定されている場合にそれを表示することです。

**手順:**

1. **ネットワーク関連のコマンドのリストを定義する。**
2. **プロキシ設定を表示する関数を作成する。**
3. **コマンドとプロキシ設定を確認する関数を作成する。**
4. **各コマンド実行前に確認を実行するために `DEBUG` トラップを設定する。**
5. **必要に応じてプロキシ設定を表示するための手動 `checkproxy` 関数を定義する。**
6. **すべての設定を `.bashrc` ファイルに追加する。**

**実装:** 以下のコードを `~/.bashrc` ファイルに追加します（存在しない場合は作成します）：

```
# ネットワーク関連のコマンドのリスト
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"
)

# プロキシ設定を表示する関数
display_proxy() {
    echo -e "  ** プロキシ設定が検出されました:**"
    [ -n "$HTTP_PROXY" ] && echo "   - HTTP_PROXY: $HTTP_PROXY"
    [ -n "$http_proxy" ] && echo "   - http_proxy: $http_proxy"
    [ -n "$HTTPS_PROXY" ] && echo "   - HTTPS_PROXY: $HTTPS_PROXY"
    [ -n "$https_proxy" ] && echo "   - https_proxy: $https_proxy"
    [ -n "$ALL_PROXY" ] && echo "   - ALL_PROXY: $ALL_PROXY"
    [ -n "$all_proxy" ] && echo "   - all_proxy: $all_proxy"
```

```
    echo ""
}

# コマンドがネットワーク関連であるか、プロキシが設定されているかを確認する関数
proxy_check() {
    local cmd
    # コマンドの最初の単語を抽出
    cmd=$(echo "$BASH_COMMAND" | awk '{print $1}')

    for network_cmd in "${network_commands[@]}"; do
        if [[ "$cmd" == "$network_cmd" ]]; then
            # プロキシ環境変数が設定されているか確認
            if [ -n "$HTTP_PROXY" ] || [ -n "$http_proxy" ] || \
               [ -n "$HTTPS_PROXY" ] || [ -n "$https_proxy" ] || \
               [ -n "$ALL_PROXY" ] || [ -n "$all_proxy" ]; then
                display_proxy
            fi
            break
        fi
    done
}

# 各コマンド実行前に `proxy_check` を実行するために DEBUG トラップを設定
trap 'proxy_check' DEBUG

# プロキシ設定を手動で確認する関数
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
}
```

**動作方法:**

- `network_commands` 配列にはネットワーク関連のコマンドがリストされています。
- `display_proxy` は設定されている場合にすべての関連するプロキシ環境変数を表示します。
- `proxy_check` は `BASH_COMMAND`（`DEBUG` トラップで利用可能）を使用して実行されるコマンドを取得し、最

初の単語を抽出し、ネットワークコマンドと一致するか確認します。プロキシ変数が設定されている場合はそれを表示します。

- `trap 'proxy_check' DEBUG` 行は、`proxy_check` が各コマンド実行前に実行されるようにします。
- `checkproxy` はプロキシ設定を手動で表示するためのもので、Git 特有のプロキシ設定を含みます。
- `.bashrc` に追加した後、Git-Bash を再起動するか `source ~/.bashrc` を実行して変更を適用します。

**使用方法:**

- ネットワークコマンド（例：`git clone`、`curl`）を実行すると、プロキシ設定が構成されている場合は、コマンド実行前に表示されます。
- `checkproxy` を実行してプロキシ設定を手動で表示します。

---

**PowerShell の場合**

PowerShell には Bash の `DEBUG` トラップの直接的な対応がないため、`PSReadLine` モジュールの `CommandValidationHandler` を使用して類似の機能を実現します。このハンドラは各コマンド実行前に実行され、ネットワークコマンドとプロキシ設定を確認することができます。

**手順:**

1. **ネットワーク関連のコマンドのリストを定義する。**
2. **プロキシ設定を表示する関数を作成する。**
3. **コマンドとプロキシ設定を確認するために `CommandValidationHandler` を設定する。**
4. **必要に応じてプロキシ設定を表示するための手動 `checkproxy` 関数を定義する。**
5. **すべての設定を PowerShell プロファイルに追加する。**

**実装:** まず、PowerShell で `$PROFILE` を実行してプロファイルファイルの場所を確認し、存在しない場合は作成します：

```
New-Item -Type File -Force $PROFILE
```

以下のコードを PowerShell プロファイル（例：`Microsoft.PowerShell_profile.ps1`）に追加します：

```
# ネットワーク関連のコマンドのリスト
$networkCommands = @(
    "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"
)

# プロキシ設定を表示する関数
```

```powershell
function Display-Proxy {
    Write-Host "  ** プロキシ設定が検出されました:**"
    if ($env:HTTP_PROXY) { Write-Host "    - HTTP_PROXY: $env:HTTP_PROXY" }
    if ($env:http_proxy) { Write-Host "    - http_proxy: $env:http_proxy" }
    if ($env:HTTPS_PROXY) { Write-Host "    - HTTPS_PROXY: $env:HTTPS_PROXY" }
    if ($env:https_proxy) { Write-Host "    - https_proxy: $env:https_proxy" }
    if ($env:ALL_PROXY) { Write-Host "    - ALL_PROXY: $env:ALL_PROXY" }
    if ($env:all_proxy) { Write-Host "    - all_proxy: $env:all_proxy" }
    Write-Host ""
}

# コマンド実行前にコマンドを確認するために CommandValidationHandler を設定
Set-PSReadLineOption -CommandValidationHandler {
    param($command)
    # コマンドの最初の単語を抽出
    $cmd = ($command -split ' ')[0]

    if ($networkCommands -contains $cmd) {
        # プロキシ環境変数が設定されているか確認
        if ($env:HTTP_PROXY -or $env:http_proxy -or $env:HTTPS_PROXY -or $env:https_proxy -or $env:ALL_PROXY -
            Display-Proxy
        }
    }
    # コマンドの実行を許可するために $true を返す
    return $true
}

# プロキシ設定を手動で確認する関数
function checkproxy {
    Write-Host "HTTP_PROXY: $env:HTTP_PROXY"
    Write-Host "HTTPS_PROXY: $env:HTTPS_PROXY"
    Write-Host "Git HTTP Proxy:"
    git config --get http.proxy
    Write-Host "Git HTTPS Proxy:"
    git config --get https.proxy
}
```

**動作方法:**

- `$networkCommands` はネットワーク関連のコマンドの配列です。

- `Display-Proxy` は設定されている場合にすべての関連するプロキシ環境変数を表示します。
- `Set-PSReadLineOption -CommandValidationHandler` は各コマンド実行前にスクリプトブロックを定義します：
  - コマンドの最初の単語を抽出します。
  - `$networkCommands` に含まれているか確認します。
  - プロキシ変数が設定されている場合は `Display-Proxy` を呼び出します。
  - コマンドの実行を許可するために `$true` を返します。
- `checkproxy` はプロキシ設定を手動で表示するためのもので、Git 特有のプロキシを含みます。
- プロファイルに追加した後、PowerShell を再起動するか `. $PROFILE` を実行して変更を適用します。

**要件:**

- `PSReadLine` モジュールが必要であり、PowerShell 5.1 以降にはデフォルトで含まれています。
- 古いバージョンを使用している場合は、PowerShell をアップグレードするか、代替方法を検討する必要があります（ここではカバーしていませんが、ほとんどのシステムは新しいバージョンを使用しています）。

**使用方法:**

- ネットワークコマンド（例：`git pull`、`curl`）を実行すると、プロキシ設定が構成されている場合は、コマンド実行前に表示されます。
- `checkproxy` を実行してプロキシ設定を手動で表示します。

---

**「ターミナル」に関する注意事項**

- 「ターミナル」が Windows Terminal を指している場合、それは Git-Bash、PowerShell、またはコマンドプロンプト（cmd.exe）などのシェルをホストするものです。
- 上記の実装は、Windows Terminal 内の Git-Bash または PowerShell セッションで動作します。
- コマンドプロンプト（cmd.exe）での同様の機能の実装は、限られたスクリプト機能のため実用的ではありません。Git-Bash または PowerShell を使用することをお勧めします。

---

**追加の考慮事項**

- **コマンドの解析:**
  - 両方の実装は、コマンドの最初の単語のみをネットワークコマンドのリストと照合します。例えば、`git clone` は `git` がリストにあるためトリガーされます。

  - 多くの場合、`bundle exec jekyll` のような複数単語のコマンドは、`bundle` がリストにあるためトリガーされます。
  - 必要に応じて、コマンドのすべての単語を確認するようにコードを修正することもできますが、これは誤検知を引き起こす可能性があり、一般的には不要です。

- **プロキシ変数:**
  - 両方の実装は、`HTTP_PROXY`、`http_proxy`、`HTTPS_PROXY`、`https_proxy`、`ALL_PROXY`、`all_proxy` を確認して、一般的な変種をカバーしています。
  - Windows では環境変数は大文字小文字を区別しないが、Unix 風の動作（特に Git-Bash）との一貫性を保つために両方のケースを確認します。

- **Git プロキシ設定:**
  - 両方のシェルの `checkproxy` 関数は、`git config --get` を使用して Git 特有のプロキシ設定を表示します。

---

## テスト

- **プロキシ変数を設定:**
  - Git-Bash: `export HTTP_PROXY=http://proxy.example.com:8080`
  - PowerShell: `$env:HTTP_PROXY = "http://proxy.example.com:8080"`

- **ネットワークコマンドを実行:**
  - `git --version`、`curl -V` などのコマンドを試してみてください。
  - プロキシ設定はコマンド出力の前に表示されるはずです。

- **`checkproxy` を使用:**
  - 両方のシェルで `checkproxy` を実行してプロキシ設定を手動で表示します。

---

この実装は、Windows Terminal 内または独立して使用するための、Git-Bash と PowerShell でのプロキシチェックを提供します。