实现 Windows 代理检查

在 Windows 中使用 Git-Bash 和 PowerShell 实现代理检查，需要在每个 shell 中设置机制，以检测网络相关命令的执行，并在配置了代理设置时显示它们。以下是 Git-Bash 和 PowerShell 的步骤和代码。提到的“终端”可能指的是托管这些 shell 的 Windows Terminal，因此我们将重点放在 Git-Bash 和 PowerShell 的实现上。

Git-Bash

Git-Bash 是 Windows 上的 Bash 模拟器，我们可以使用 `DEBUG` 陷阱在每次命令执行之前运行一个函数。目标是检查命令是否与网络相关，如果代理设置已配置，则显示它们。

步骤：

1. **定义网络相关命令的列表。**
2. **创建一个函数来显示代理设置。**
3. **创建一个函数来检查命令和代理设置。**
4. **设置 `DEBUG` 陷阱在每个命令之前运行检查。**
5. **定义一个手动 `checkproxy` 函数以按需显示代理设置。**
6. **将所有配置添加到 `.bashrc` 文件中。**

实现： 将以下代码添加到您的 `~/.bashrc` 文件（如果不存在则创建它）：

```
# 网络相关命令列表
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"
)

# 函数显示代理设置
display_proxy() {
    echo -e "  ** 检测到代理设置：**"
    [ -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 ""
}
```

```
# 函数检查命令是否与网络相关并设置代理
proxy_check() {
    local cmd
    # 提取命令的第一个单词
    cmd=$(echo "$BASH_COMMAND" | awk '{print $1}')

    for network_cmd in "${network_commands[@]}"; do
        if [[ "$cmd" == "$network_cmd" ]]; then
            # 检查是否设置了任何代理环境变量
            if [ -n "$HTTP_PROXY" ] || [ -n "$http_proxy" ] || \
               [ -n "$HTTPS_PROXY" ] || [ -n "$https_proxy" ] || \
               [ -n "$ALL_PROXY" ] || [ -n "$all_proxy" ]; then
                display_proxy
            fi
            break
        fi
    done
}

# 设置 DEBUG 陷阱在每个命令之前运行 proxy_check
trap 'proxy_check' DEBUG

# 函数手动检查代理设置
checkproxy() {
    echo "HTTP_PROXY: $HTTP_PROXY"
    echo "HTTPS_PROXY: $HTTPS_PROXY"
    echo "Git HTTP 代理："
    git config --get http.proxy
    echo "Git HTTPS 代理："
    git config --get https.proxy
}
```

工作原理：

- `network_commands` 数组列出了与网络相关的命令。
- `display_proxy` 显示所有相关的代理环境变量（如果已设置）。
- `proxy_check` 使用 `BASH_COMMAND`（在 `DEBUG` 陷阱中可用）获取正在执行的命令，提取第一个单词，并检查它是否与网络命令匹配。如果设置了代理变量，则显示它们。
- `trap 'proxy_check' DEBUG` 行确保在每个命令之前运行 `proxy_check`。
- `checkproxy` 允许您手动查看代理设置，包括特定于 Git 的代理配置。

- 添加到 `.bashrc` 后，重新启动 Git-Bash 或运行 `source ~/.bashrc` 以应用更改。

使用：

- 当您运行网络命令（例如 `git clone`、`curl`）时，如果配置了代理设置，它们将在命令执行之前显示。
- 运行 `checkproxy` 以手动查看代理设置。

PowerShell

PowerShell 没有直接等同于 Bash 的 `DEBUG` 陷阱，但我们可以使用 `PSReadLine` 模块的 `CommandValidationHandler` 来实现类似的功能。该处理程序在每个命令之前运行，允许我们检查网络命令和代理设置。

步骤：

1. **定义网络相关命令的列表。**
2. **创建一个函数来显示代理设置。**
3. **设置 `CommandValidationHandler` 来检查命令和代理设置。**
4. **定义一个手动 `checkproxy` 函数以按需显示代理设置。**
5. **将所有配置添加到您的 PowerShell 个人资料中。**

实现： 首先，通过在 PowerShell 中运行 `$PROFILE` 来定位您的 PowerShell 个人资料文件。如果不存在，请创建它：

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

将以下代码添加到您的 PowerShell 个人资料（例如 `Microsoft.PowerShell_profile.ps1`）：

```
# 网络相关命令列表
$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"
)

# 函数显示代理设置
function Display-Proxy {
    Write-Host "  ** 检测到代理设置：**"
    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 ""
}

# 设置 CommandValidationHandler 在执行命令之前检查命令
Set-PSReadLineOption -CommandValidationHandler {
    param($command)
    # 提取命令的第一个单词
    $cmd = ($command -split ' ')[0]

    if ($networkCommands -contains $cmd) {
        # 检查是否设置了任何代理环境变量
        if ($env:HTTP_PROXY -or $env:http_proxy -or $env:HTTPS_PROXY -or $env:https_proxy -or $env:ALL_PROXY -
            Display-Proxy
        }
    }
    # 总是返回 true 以允许命令执行
    return $true
}

# 函数手动检查代理设置
function checkproxy {
    Write-Host "HTTP_PROXY: $env:HTTP_PROXY"
    Write-Host "HTTPS_PROXY: $env:HTTPS_PROXY"
    Write-Host "Git HTTP 代理: "
    git config --get http.proxy
    Write-Host "Git HTTPS 代理: "
    git config --get https.proxy
}
```

工作原理：

- `$networkCommands` 是一个与网络相关的命令数组。
- `Display-Proxy` 显示所有相关的代理环境变量（如果已设置）。
- `Set-PSReadLineOption -CommandValidationHandler` 定义一个脚本块，在每个命令之前运行：
  - 提取命令的第一个单词。
  - 检查它是否在 `$networkCommands` 中。
  - 如果设置了代理变量，则调用 `Display-Proxy`。
  - 返回 `$true` 以确保命令执行。

- `checkproxy` 允许手动查看代理设置，包括特定于 Git 的代理。
- 添加到个人资料后，重新启动 PowerShell 或运行 `. $PROFILE` 以应用更改。

要求：

- 需要 `PSReadLine` 模块，该模块默认包含在 PowerShell 5.1 及更高版本中。
- 如果使用较旧版本，可能需要升级 PowerShell 或找到替代方法（此处未涵盖，因为大多数系统使用较新版本）。

使用：

- 当您运行网络命令（例如 `git pull`、`curl`）时，如果配置了代理设置，它们将在命令执行之前显示。
- 运行 `checkproxy` 以手动查看代理设置。

关于“终端”的说明

- 如果“终端”指的是 Windows Terminal，它只是 Git-Bash、PowerShell 或命令提示符（cmd.exe）的宿主。
- 上述实现在 Windows Terminal 中的 Git-Bash 或 PowerShell 会话中工作。
- 在命令提示符（cmd.exe）中实现类似功能不切实际，因为其脚本功能有限。建议使用 Git-Bash 或 PowerShell 代替。

其他考虑

- **命令解析：**
  - 两种实现都只检查命令的第一个单词与网络命令列表的匹配。例如，`git clone` 会触发，因为 `git` 在列表中。
  - 多单词命令（例如 `bundle exec jekyll`）将触发，如果 `bundle` 在列表中，这对于大多数情况是足够的。
  - 如果需要，可以修改代码以检查命令中的所有单词，但这可能导致误报，通常是不必要的。
- **代理变量：**
  - 两种实现都检查 `HTTP_PROXY`、`http_proxy`、`HTTPS_PROXY`、`https_proxy`、`ALL_PROXY` 和 `all_proxy` 以涵盖常见变体。
  - 在 Windows 中，环境变量不区分大小写，但我们检查两种情况以与类 Unix 行为一致（特别是在 Git-Bash 中）。
- **Git 代理设置：**
  - 两个 shell 中的 `checkproxy` 函数使用 `git config --get` 显示特定于 Git 的代理设置。

测试

- **设置代理变量：**
  - 在 Git-Bash：`export HTTP_PROXY=http://proxy.example.com:8080`
  - 在 PowerShell：`$env:HTTP_PROXY = "http://proxy.example.com:8080"`
- **运行网络命令：**
  - 尝试命令如 `git --version`、`curl -V` 等。
  - 代理设置应在命令输出之前显示。
- **使用 `checkproxy`：**
  - 在任何 shell 中运行 `checkproxy` 以手动查看代理设置。

此实现在 Git-Bash 和 PowerShell 中提供了强大的代理检查，适用于在 Windows Terminal 或独立使用。