prplOS Development Environment Setup Guide

Table of Contents

1. Recommended Operating System {#recommended-os}

Primary Recommendation: **Ubuntu 22.04 LTS or Ubuntu 20.04 LTS**

Why Ubuntu?

- Native Linux environment (OpenWrt/prplOS is designed for Linux)
- Best compatibility with build tools
- Fastest compilation times
- Full feature support
- Extensive community support

Alternative Options (in order of preference):

1. **Debian 11/12** - Very similar to Ubuntu
2. **Fedora 37+** - Good alternative, may need package name adjustments
3. **WSL2 on Windows** - Best Windows option
4. **MSYS2 UCRT64** - Windows alternative with limitations
5. **macOS** - Possible but requires additional setup

2. System Requirements {#system-requirements}

Minimum Requirements:

- **CPU**: 64-bit processor (x86_64), 2+ cores
- **RAM**: 4GB (8GB recommended)
- **Storage**: 50GB free space (100GB recommended for multiple builds)
- **Internet**: Stable connection for downloading sources

Recommended Requirements:

- **CPU**: 4+ cores (8+ for faster builds)
- **RAM**: 16GB or more
- **Storage**: 200GB+ SSD
- **OS**: Ubuntu 22.04 LTS

3. Ubuntu Setup (Recommended) {#ubuntu-setup}

Step 1: Update System

```bash
sudo apt update && sudo apt upgrade -y
```

Step 2: Install Essential Packages

bash

```bash
# Core build tools
sudo apt install -y \
    build-essential \
    gcc \
    g++ \
    make \
    cmake \
    automake \
    autoconf \
    libtool

# Version control
sudo apt install -y \
    git \
    git-lfs \
    subversion \
    mercurial

# Patch management tools
sudo apt install -y \
    quilt \
    patch \
    diffutils \
    patchutils

# Development libraries
sudo apt install -y \
    libncurses5-dev \
    libncursesw5-dev \
    zlib1g-dev \
    libssl-dev \
    libelf-dev \
    liblzma-dev \
    libbz2-dev

# Python and dependencies
sudo apt install -y \
    python3 \
    python3-dev \
    python3-pip \
    python3-setuptools \
    python3-distutils \
    python3-venv
```

```
# Additional tools
sudo apt install -y \
    gawk \
    wget \
    curl \
    file \
    unzip \
    rsync \
    bc \
    time \
    gettext \
    libtext-csv-perl \
    xsltproc \
    libxml2-utils

# Optional but recommended
sudo apt install -y \
    ccache \
    ninja-build \
    tmux \
    htop \
    iotop \
    sysstat
```

Step 3: Install Python Packages

bash

```
pip3 install --user \
    matplotlib \
    pandas \
    numpy \
    seaborn \
    psutil \
    requests
```

Step 4: Configure Quilt

bash

```bash
cat > ~/.quiltrc << 'EOF'
QUILT_DIFF_ARGS="--no-timestamps --no-index -p ab --color=auto"
QUILT_REFRESH_ARGS="--no-timestamps --no-index -p ab"
QUILT_SERIES_ARGS="--color=auto"
QUILT_PATCH_OPTS="--unified"
QUILT_DIFF_OPTS="-p"
EDITOR="nano"
EOF
```

Step 5: Setup Build Environment

bash

```bash
# Create workspace
mkdir -p ~/prplos-workspace
cd ~/prplos-workspace

# Clone scripts repository (assuming you have them)
git clone <your-patch-management-repo> patch-management

# Set environment variables
echo 'export PRPLOS_ROOT="$HOME/prplos-workspace/prplos"' >> ~/.bashrc
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

Step 6: Clone and Setup prplOS

bash

```bash
# Clone prplOS
git clone https://gitlab.com/prpl-foundation/prplos/prplos.git
cd prplos

# Update and install feeds
./scripts/feeds update -a
./scripts/feeds install -a

# Configure
make menuconfig  # Or use a default config
```

4. Windows Setup Options {#windows-setup}

Option A: WSL2 Setup (Recommended for Windows) {#wsl2-setup}

Prerequisites:

- Windows 10 version 2004+ or Windows 11
- Virtualization enabled in BIOS

Step 1: Install WSL2

```
powershell

# Run in PowerShell as Administrator
wsl --install -d Ubuntu-22.04

# Set WSL2 as default
wsl --set-default-version 2

# Verify installation
wsl --list --verbose
```

Step 2: Configure WSL2

```
bash

# Inside WSL2 Ubuntu terminal
# Increase WSL2 memory (create .wslconfig in Windows user home)
cat > /mnt/c/Users/$USER/.wslconfig << EOF
[wsl2]
memory=8GB
processors=4
swap=8GB
EOF
```

Step 3: Follow Ubuntu Setup

Once in WSL2, follow the complete Ubuntu setup instructions above.

WSL2-Specific Adjustments:

```bash
# Better performance by using Linux filesystem
cd ~  # Work in Linux filesystem, not /mnt/c/

# Fix potential clock skew issues
sudo hwclock -s

# Install WSL utilities
sudo apt install -y wslu
```

Option B: MSYS2 UCRT64 Setup {#msys2-setup}

Note: MSYS2 has limitations for OpenWrt/prplOS development. Some features may not work properly.

Step 1: Install MSYS2

1. Download from: https://www.msys2.org/
2. Install to default location (C:\msys64)
3. Run MSYS2 UCRT64 terminal

Step 2: Update MSYS2

```bash
# In MSYS2 UCRT64 terminal
pacman -Syu
# Close terminal when prompted, then reopen and run:
pacman -Su
```

Step 3: Install Required Packages

bash

```
# Development tools
pacman -S --needed \
    base-devel \
    mingw-w64-ucrt-x86_64-toolchain \
    mingw-w64-ucrt-x86_64-cmake \
    mingw-w64-ucrt-x86_64-ninja

# Version control
pacman -S --needed \
    git \
    subversion \
    mercurial

# Patch tools
pacman -S --needed \
    quilt \
    patch \
    diffutils \
    patchutils

# Python and dependencies
pacman -S --needed \
    mingw-w64-ucrt-x86_64-python \
    mingw-w64-ucrt-x86_64-python-pip \
    mingw-w64-ucrt-x86_64-python-numpy \
    mingw-w64-ucrt-x86_64-python-pandas \
    mingw-w64-ucrt-x86_64-python-matplotlib

# Additional tools
pacman -S --needed \
    autoconf \
    automake \
    libtool \
    make \
    wget \
    curl \
    rsync \
    bc \
    time \
    unzip \
    ncurses-devel \
    zlib-devel \
    openssl-devel
```

Step 4: Configure MSYS2 Environment

```bash
# Add to ~/.bashrc
echo 'export MSYSTEM=UCRT64' >> ~/.bashrc
echo 'export PATH="/ucrt64/bin:$PATH"' >> ~/.bashrc

# Configure git for Windows line endings
git config --global core.autocrlf false
git config --global core.eol lf

# Setup quilt
cat > ~/.quiltrc << 'EOF'
QUILT_DIFF_ARGS="--no-timestamps --no-index -p ab"
QUILT_REFRESH_ARGS="--no-timestamps --no-index -p ab"
QUILT_SERIES_ARGS=""
QUILT_PATCH_OPTS="--unified"
QUILT_DIFF_OPTS="-p"
EDITOR="nano"
EOF
```

MSYS2 Limitations:

- Case-sensitive filesystem issues
- Symbolic link limitations
- Slower build performance
- Some Linux-specific features may not work
- Path translation issues between Windows and MSYS2

Option C: Native Windows (Limited) {#native-windows}

Not Recommended: Very limited functionality, only for viewing/editing patches.

Tools for Native Windows:

1. **Git for Windows**: https://git-scm.com/download/win
2. **Python**: https://www.python.org/downloads/windows/
3. **Visual Studio Code**: For editing

4. **TortoiseGit**: GUI for Git operations

5. Common Prerequisites Summary {#common-prerequisites}

Core Requirements Table

Component	Ubuntu	WSL2	MSYS2 UCRT64	Purpose
gcc/g++	✓	✓	✓ (mingw)	Compilation
make	✓	✓	✓	Build automation
git	✓	✓	✓	Version control
quilt	✓	✓	✓	Patch management
python3	✓	✓	✓	Scripting/Analysis
bc	✓	✓	✓	Calculations
patch	✓	✓	✓	Applying patches
wget/curl	✓	✓	✓	Downloading
ncurses	✓	✓	✓	Menu config

Python Package Requirements

```txt
# requirements.txt
matplotlib>=3.5.0
pandas>=1.4.0
numpy>=1.22.0
seaborn>=0.11.0
psutil>=5.9.0
requests>=2.27.0
```

Install with:

```bash
pip3 install -r requirements.txt
```

6. Verification Scripts {#verification}

Create verification script

bash

```
cat > verify_environment.sh << 'EOF'
#!/bin/bash

echo "=== prplOS Development Environment Verification ==="
echo

# Colors
GREEN='\033[0;32m'
RED='\033[0;31m'
YELLOW='\033[1;33m'
NC='\033[0m'

# Check function
check_command() {
    if command -v $1 &> /dev/null; then
        version=$($1 --version 2>&1 | head -n1)
        echo -e "${GREEN}✓${NC} $1: $version"
        return 0
    else
        echo -e "${RED}✗${NC} $1: NOT FOUND"
        return 1
    fi
}

check_python_package() {
    if python3 -c "import $1" 2>/dev/null; then
        version=$(python3 -c "import $1; print($1.__version__)" 2>/dev/null || echo "installed"
        echo -e "${GREEN}✓${NC} Python $1: $version"
        return 0
    else
        echo -e "${RED}✗${NC} Python $1: NOT INSTALLED"
        return 1
    fi
}

# System info
echo "System Information:"
echo "OS: $(uname -s) $(uname -r)"
echo "Architecture: $(uname -m)"
echo "CPU Cores: $(nproc)"
echo "Memory: $(free -h | awk '/^Mem:/ {print $2}')"
echo
```

```
# Check essential tools
echo "Essential Tools:"
failed=0
for cmd in gcc g++ make git quilt patch python3 bc wget curl; do
    check_command $cmd || ((failed++))
done
echo

# Check Python packages
echo "Python Packages:"
for pkg in matplotlib pandas numpy seaborn; do
    check_python_package $pkg || ((failed++))
done
echo

# Check disk space
echo "Disk Space:"
df -h . | tail -1
echo

# Check quilt configuration
echo "Quilt Configuration:"
if [ -f ~/.quiltrc ]; then
    echo -e "${GREEN}✓${NC} ~/.quiltrc exists"
else
    echo -e "${YELLOW}!${NC} ~/.quiltrc not found (will use defaults)"
fi
echo

# Summary
if [ $failed -eq 0 ]; then
    echo -e "${GREEN}All checks passed! Environment is ready for prplOS development.${NC}"
    exit 0
else
    echo -e "${RED}$failed checks failed. Please install missing components.${NC}"
    exit 1
fi
EOF

chmod +x verify_environment.sh
./verify_environment.sh
```

Quick Setup Script

bash

```
cat > quick_setup.sh << 'EOF'
#!/bin/bash

# Detect OS
if [ -f /etc/os-release ]; then
    . /etc/os-release
    OS=$ID
    VER=$VERSION_ID
else
    echo "Cannot detect OS"
    exit 1
fi

echo "Detected OS: $OS $VER"

case $OS in
    ubuntu|debian)
        echo "Installing packages for Ubuntu/Debian..."
        sudo apt update
        sudo apt install -y \
            build-essential git quilt python3 python3-pip \
            bc wget curl patch diffutils time \
            libncurses-dev zlib1g-dev libssl-dev
        ;;
    fedora|rhel|centos)
        echo "Installing packages for Fedora/RHEL..."
        sudo dnf install -y \
            @development-tools git quilt python3 python3-pip \
            bc wget curl patch diffutils time \
            ncurses-devel zlib-devel openssl-devel
        ;;
    *)
        echo "Unsupported OS: $OS"
        echo "Please install packages manually"
        exit 1
        ;;
esac

# Install Python packages
pip3 install --user matplotlib pandas numpy seaborn

# Setup quilt
if [ ! -f ~/.quiltrc ]; then
```

```
    cat > ~/.quiltrc << 'QUILTRC'
QUILT_DIFF_ARGS="--no-timestamps --no-index -p ab --color=auto"
QUILT_REFRESH_ARGS="--no-timestamps --no-index -p ab"
QUILT_SERIES_ARGS="--color=auto"
QUILT_PATCH_OPTS="--unified"
QUILT_DIFF_OPTS="-p"
EDITOR="nano"
QUILTRC
fi

echo "Setup complete! Run ./verify_environment.sh to check."
EOF

chmod +x quick_setup.sh
```

7. Troubleshooting {#troubleshooting}

Common Issues and Solutions

Ubuntu/Debian Issues

Problem: Package not found

```
bash

# Solution: Update package lists
sudo apt update
sudo apt install -y software-properties-common
```

Problem: Python package installation fails

```
bash

# Solution: Use virtual environment
python3 -m venv prplos-env
source prplos-env/bin/activate
pip install matplotlib pandas numpy seaborn
```

WSL2 Issues

Problem: Clock skew

```bash
# Solution: Sync time
sudo hwclock -s
# Or
sudo ntpdate -s time.nist.gov
```

Problem: Out of memory

```bash
# Solution: Increase WSL2 memory in .wslconfig
# Edit C:\Users\YourUsername\.wslconfig
[wsl2]
memory=8GB
swap=8GB
```

Problem: Slow I/O performance

```bash
# Solution: Work in Linux filesystem
cd ~  # Don't use /mnt/c/
```

MSYS2 Issues

Problem: Command not found

```bash
# Solution: Ensure correct terminal (UCRT64)
# Check $MSYSTEM variable
echo $MSYSTEM  # Should be UCRT64
```

Problem: Path issues

```bash
# Solution: Use MSYS2 path format
# Windows: C:\Users\Name\file
# MSYS2: /c/Users/Name/file
```

Problem: Line ending issues

```bash
# Solution: Configure git
git config --global core.autocrlf false
git config --global core.eol lf

# Convert existing files
dos2unix *.patch
```

Performance Optimization Tips

For All Platforms:

```bash
# Enable ccache
export USE_CCACHE=1
export CCACHE_DIR=~/.ccache
ccache -M 10G  # Set 10GB cache

# Parallel builds
export MAKEFLAGS="-j$(nproc)"

# Use tmpfs for build (Linux/WSL2)
mkdir -p /tmp/build
export TMPDIR=/tmp/build
```

Platform-Specific:

Ubuntu:

```bash
# Install on SSD if possible
# Enable zram for more memory
sudo apt install zram-config
```

WSL2:

```bash
# Disable Windows Defender for WSL2 folders
# Add exclusion in Windows Defender settings
```

MSYS2:

```bash
# Disable antivirus real-time scanning for MSYS2 folder
# Use RAM disk for temporary files
```

Minimum Test

To verify basic functionality:

```bash
# Create test directory
mkdir -p ~/prplos-test
cd ~/prplos-test

# Test git
git init test-repo

# Test quilt
mkdir patches
echo "test" > file.txt
quilt new test.patch
quilt add file.txt
echo "modified" > file.txt
quilt refresh

# Test Python
python3 -c "import matplotlib, pandas, numpy; print('Python packages OK')"

# Test compilation
echo '#include <stdio.h>
int main() { printf("Build test OK\\n"); return 0; }' > test.c
gcc test.c -o test && ./test

echo "Basic functionality verified!"
```

Summary

Recommended Setup Priority:

1. **Ubuntu 22.04 LTS** - Native Linux, best performance
2. **WSL2 on Windows** - Good Windows option, near-native performance

3. **MSYS2 UCRT64** - Windows alternative with limitations
4. **Native Windows** - Only for viewing/editing, not building

Quick Start Commands:

```bash
# Ubuntu/WSL2
./quick_setup.sh
./verify_environment.sh

# Start using the automation suite
./prplos-patch-automation-suite.sh setup
./prplos-patch-automation-suite.sh benchmark
```

For production use, Ubuntu or WSL2 is strongly recommended due to better compatibility and performance with the OpenWrt/prplOS build system.