Complete Guide: Compiling Festival 1.96 on Modern Linux and Creating a Debian Package

This comprehensive guide documents the complete process for successfully compiling Festival Speech Synthesis System version 1.96 on modern Linux systems with current GCC compilers, and packaging it as a Debian package with high-quality Nitech HTS voices.

Table of Contents

Prerequisites

Install required development libraries:

```bash
sudo apt-get update
sudo apt-get install build-essential libncurses5-dev libtinfo-dev alsa-utils
```

Step 1: Download Required Files

Download all necessary files from `http://festvox.org/packed/festival/1.96/`:

```bash
mkdir -p ~/app_installs/festival/196/build
cd ~/app_installs/festival/196/build

# Download core components
wget http://festvox.org/packed/festival/1.96/speech_tools-1.2.96-beta.tar.gz
wget http://festvox.org/packed/festival/1.96/festival-1.96-beta.tar.gz

# Download lexicons (required)
wget http://festvox.org/packed/festival/1.96/festlex_CMU.tar.gz
wget http://festvox.org/packed/festival/1.96/festlex_POSLEX.tar.gz

# Download at least one voice
wget http://festvox.org/packed/festival/1.96/festvox_kallpc16k.tar.gz
```

Alternative: To download all files automatically:

```bash
wget -r -np -nH --cut-dirs=3 -R "index.html*" -e robots=off --wait=1 \
  http://festvox.org/packed/festival/1.96/
```

Step 2: Extract All Archives

Extract everything in the same parent directory:

```bash
cd ~/app_installs/festival/196/build

tar xzf speech_tools-1.2.96-beta.tar.gz
tar xzf festival-1.96-beta.tar.gz
tar xzf festlex_CMU.tar.gz
tar xzf festlex_POSLEX.tar.gz
tar xzf festvox_kallpc16k.tar.gz
```

The lexicons and voices will automatically extract into `festival/lib/`.

Step 3: Modify GCC Configuration for Speech Tools

Modern linkers require special handling for duplicate symbols in old code.

Edit the GCC defaults file:

```bash
cd ~/app_installs/festival/196/build/speech_tools
nano config/compilers/gcc_defaults.mak
```

Add `-Wl,--allow-multiple-definition` to **all** `*_LINKFLAGS` lines. Find these lines and modify them:

```make
DEBUG_LINKFLAGS = -g -Wl,--allow-multiple-definition
WARN_LINKFLAGS = -Wall -Wl,--allow-multiple-definition
VERBOSE_LINKFLAGS = -Wl,--allow-multiple-definition
OPTIMISE_LINKFLAGS = -O$(OPTIMISE) -Wl,--allow-multiple-definition
PROFILE_prof_LINKFLAGS = -p -Wl,--allow-multiple-definition
PROFILE_gprof_LINKFLAGS = -pg -Wl,--allow-multiple-definition
SHARED_LINKFLAGS = -fno-shared-data -Wl,--allow-multiple-definition
STATIC_LINKFLAGS = -static -Wl,--allow-multiple-definition
```

Save and exit (Ctrl+X, Y, Enter).

Step 4: Compile Speech Tools Library

Configure and compile Speech Tools with compatibility flags:

```bash
cd ~/app_installs/festival/196/build/speech_tools
./configure
make CXXFLAGS="-O0 -g -Wall -fpermissive -std=c++98 -D_GLIBCXX_USE_CXX11_ABI=0 -DSUPP
```

Key compiler flags explained:

- `-fpermissive`: Downgrades C++ errors to warnings for old code
- `-std=c++98`: Uses C++98 standard that Festival was designed for
- `-D_GLIBCXX_USE_CXX11_ABI=0`: Forces old C++ ABI for compatibility with modern systems (CRITICAL)
- `-DSUPPORT_EDITLINE`: Enables command-line editing features

Expected result: Compilation should complete with a "Remove Links:" message showing successful build.

Step 5: Modify GCC Configuration for Festival

Festival needs the same linker flags as Speech Tools. First, run configure to generate the config files:

```bash
cd ~/app_installs/festival/196/build/festival
./configure
```

Then, if `config/compilers/gcc_defaults.mak` exists, edit it:

```bash
nano config/compilers/gcc_defaults.mak
```

Add the same `-Wl,--allow-multiple-definition` to all `*_LINKFLAGS` lines as in Step 3.

Note: If this file doesn't exist, Festival will inherit the compiler settings from Speech Tools, which is acceptable.

Step 6: Configure Festival Installation Paths

CRITICAL STEP: By default, Festival compiles with hardcoded paths pointing to the build directory. We need to change this so the installed Festival looks for libraries in `/usr/local/share/festival/`.

Edit the project configuration:

```bash
cd ~/app_installs/festival/196/build/festival
nano config/project.mak
```

Find the line:

```make
FTLIBDIR = $(FESTIVAL_HOME)/lib
```

Comment it out and add the installation path:

```make
#FTLIBDIR = $(FESTIVAL_HOME)/lib
FTLIBDIR = /usr/local/share/festival
```

Alternative method (more robust):

```bash
cd ~/app_installs/festival/196/build/festival
sed -i '/^FTLIBDIR = \$(FESTIVAL_HOME)\/lib$/s/^/#/' config/project.mak
echo "FTLIBDIR = /usr/local/share/festival" >> config/project.mak
```

Verify the change:

```bash
grep FTLIBDIR config/project.mak
```

Should show:

```
#FTLIBDIR = $(FESTIVAL_HOME)/lib
FTLIBDIR = /usr/local/share/festival
```

Why this matters: This tells Festival to look for its library files in `/usr/local/share/festival/` at runtime, instead of looking in your build directory. Without this change, Festival will only work from the build directory.

Save and exit.

Step 7: Compile Festival

Compile Festival with matching flags:

```bash
cd ~/app_installs/festival/196/build/festival
make CXXFLAGS="-O0 -g -Wall -fpermissive -std=c++98 -D_GLIBCXX_USE_CXX11_ABI=0"
```

Important: Use the same compiler flags as Speech Tools for ABI compatibility. The `-D_GLIBCXX_USE_CXX11_ABI=0` flag is absolutely critical.

Step 8: Test Festival

Test that Festival compiles and runs from the build directory:

```bash
cd ~/app_installs/festival/196/build/festival
./bin/festival
```

At the Festival prompt:

```
scheme

festival> (SayText "Hello world, Festival is working!")
```

You'll see "Linux: can't open /dev/dsp" - this is expected and normal. We'll configure ALSA audio in the package.

Verify the path is correct:

```
scheme

festival> (car load-path)
```

This should show `/usr/local/share/festival/...` NOT your build directory path. If it shows your build directory, go back to Step 6 and ensure FTLIBDIR was set correctly.

Exit Festival:

```
scheme

festival> (quit)
```

Step 9: Download and Install Nitech HTS Voices

The Nitech HTS voices are high-quality voices that work specifically with Festival 1.96.

Download the Nitech Voices

Using the mirrored archive (recommended):

```
bash

cd ~/app_installs/festival/196/build
wget http://erewhon.superkuh.com/nitech_voices_for_festival_196.tar.gz
tar xzf nitech_voices_for_festival_196.tar.gz
```

This extracts individual voice archive files. Now extract all of them:

```
bash

# Extract all voice archives
for f in festvox_nitech_us_*_arctic_hts-2.1.tar.bz2; do
    tar xjf "$f"
done
```

After extraction, you'll have a `lib/` directory structure containing all the voices and `hts.scm`.

Install Voices to System Location (Optional for Testing)

```bash
cd ~/app_installs/festival/196/build

# Create voice directory
sudo mkdir -p /usr/local/share/festival/voices/us

# Copy all Nitech voices from the lib directory
sudo cp -r lib/voices/us/* /usr/local/share/festival/voices/us/

# Copy the HTS engine support file
sudo cp lib/hts.scm /usr/local/share/festival/hts.scm
```

Note: This step is optional if you're going straight to creating the DEB package, as the package will include these files.

Step 10: Create Debian Package

Now we'll package everything into a distributable `.deb` file.

Create Package Directory Structure

```bash
cd ~/app_installs/festival/196/build

# Create directory structure
mkdir -p festival-1.96-deb/DEBIAN
mkdir -p festival-1.96-deb/usr/local/bin
mkdir -p festival-1.96-deb/usr/local/share/festival/voices/us
mkdir -p festival-1.96-deb/usr/local/share/doc/festival
mkdir -p festival-1.96-deb/etc/festival
```

Copy Festival Binaries

```bash
# Copy Festival binaries
cp festival/bin/festival festival-1.96-deb/usr/local/bin/
cp festival/bin/festival_client festival-1.96-deb/usr/local/bin/
cp festival/bin/text2wave festival-1.96-deb/usr/local/bin/

# Copy Speech Tools utilities
cp speech_tools/bin/ch_wave festival-1.96-deb/usr/local/bin/
cp speech_tools/bin/ch_track festival-1.96-deb/usr/local/bin/
cp speech_tools/bin/wagon festival-1.96-deb/usr/local/bin/
```

Copy Festival Libraries and Data

```bash
# Copy Festival core libraries
cp -r festival/lib/* festival-1.96-deb/usr/local/share/festival/

# Copy Nitech voices from the extracted lib directory
cp -r lib/voices/us/* festival-1.96-deb/usr/local/share/festival/voices/us/

# Copy HTS engine support
cp lib/hts.scm festival-1.96-deb/usr/local/share/festival/
```

Copy Documentation

```bash
cp festival/README festival-1.96-deb/usr/local/share/doc/festival/ 2>/dev/null || true
cp festival/ACKNOWLEDGMENTS festival-1.96-deb/usr/local/share/doc/festival/ 2>/dev/null || true
cp festival/COPYING festival-1.96-deb/usr/local/share/doc/festival/ 2>/dev/null || true
```

Create System-Wide Configuration

Create the configuration file that Festival will actually load:

```bash
cat > festival-1.96-deb/usr/local/share/festival/siteinit.scm << 'EOF'
;;; System-wide Festival configuration

; Use ALSA for audio output
(Parameter.set 'Audio_Method 'Audio_Command)
(Parameter.set 'Audio_Command "aplay -q -c 1 -t raw -f s16 -r $SR $FILE")

; Set default voice to SLT (high-quality female voice)
(set! voice_default 'voice_nitech_us_slt_arctic_hts)
EOF
```

Also create a reference copy in `/etc/festival/`:

```bash
cat > festival-1.96-deb/etc/festival/siteinit.scm << 'EOF'
;;; System-wide Festival configuration
;;; Note: The active configuration is in /usr/local/share/festival/siteinit.scm
;;; This file is for reference only.

; Use ALSA for audio output
(Parameter.set 'Audio_Method 'Audio_Command)
(Parameter.set 'Audio_Command "aplay -q -c 1 -t raw -f s16 -r $SR $FILE")

; Set default voice to SLT (high-quality female voice)
(set! voice_default 'voice_nitech_us_slt_arctic_hts)
EOF
```

Create Control File

Detect your architecture and create the control file:

```bash
ARCH=$(dpkg --print-architecture)

cat > festival-1.96-deb/DEBIAN/control << EOF
Package: festival
Version: 1.96-nitech1
Section: sound
Priority: optional
Architecture: $ARCH
Depends: libc6 (>= 2.31), libstdc++6 (>= 10), libncurses6, libtinfo6, alsa-utils
Maintainer: Your Name <your.email@example.com>
Description: Festival Speech Synthesis System with Nitech HTS Voices
 Festival is a general multi-lingual speech synthesis system developed
 at CSTR (Centre for Speech Technology Research). It offers a full text
 to speech system with various APIs, as well as an environment for
 development and research of speech synthesis techniques.
 .
 This is version 1.96 (July 2004) compiled for modern Linux systems
 with GCC compatibility patches.
 .
 This package includes:
  - Festival speech synthesis engine
  - Speech Tools utilities
  - CMU and POSLEX lexicons
  - Nitech HTS voices (high-quality parametric synthesis)
  - US English diphone voices
  - Pre-configured ALSA audio output
 .
 Available voices include:
  - nitech_us_slt_arctic_hts (female, high quality, default)
  - nitech_us_awb_arctic_hts (male)
  - nitech_us_bdl_arctic_hts (male)
  - nitech_us_clb_arctic_hts (female)
  - nitech_us_rms_arctic_hts (male)
  - nitech_us_jmk_arctic_hts (male)
EOF
```

Note: Change the Maintainer field to your actual name and email.

Create Post-Install Script

This script ensures the `audsp` binary has execute permissions (critical for `--tts` mode):

```bash
cat > festival-1.96-deb/DEBIAN/postinst << 'EOF'
#!/bin/bash
set -e

# Ensure audsp binary is executable (critical for --tts mode)
find /usr/local/share/festival -type f -name "audsp" -exec chmod 755 {} \; 2>/dev/null || true

echo ""
echo "Festival 1.96 with Nitech HTS voices has been installed."
echo ""
echo "Quick tests:"
echo "  echo 'Hello world' | festival --tts"
echo "  echo \"Hello world\" | text2wave | aplay"
echo "  echo '(SayText \"Hello world\")' | festival"
echo ""
echo "Convert text to WAV file:"
echo "  echo \"Hello world\" | text2wave -o output.wav"
echo ""
echo "Interactive mode:"
echo "  festival"
echo "  festival> (SayText \"Hello world\")"
echo ""
echo "Default voice: nitech_us_slt_arctic_hts (female)"
echo "Configuration: /usr/local/share/festival/siteinit.scm"
echo ""

exit 0
EOF

chmod 755 festival-1.96-deb/DEBIAN/postinst
```

Set Correct Permissions

This is critical - improper permissions will cause the `--tts` mode to fail:

```bash
# Set ownership to root (required for proper package installation)
sudo chown -R root:root festival-1.96-deb/

# Ensure main binaries are executable
sudo chmod 755 festival-1.96-deb/usr/local/bin/*

# Make sure audsp binary is executable (CRITICAL for --tts mode)
sudo find festival-1.96-deb/usr/local/share/festival -type f -name "audsp" -exec chmod 755 {} \;

# Make sure any other binaries in festival lib are executable
sudo find festival-1.96-deb/usr/local/share/festival/bin -type f -exec chmod 755 {} \; 2>/dev/null || t

# Set standard directory and file permissions
sudo find festival-1.96-deb/usr/local/share/festival -type d -exec chmod 755 {} \;
sudo find festival-1.96-deb/usr/local/share/festival -type f -exec chmod 644 {} \;

# Re-apply executable permissions to binaries (in case the above made them non-executable)
sudo find festival-1.96-deb/usr/local/share/festival -type f -name "audsp" -exec chmod 755 {} \;
sudo find festival-1.96-deb/usr/local/share/festival/bin -type f -exec chmod 755 {} \; 2>/dev/null || t
```

Why the audsp permissions matter: Festival's `--tts` mode uses the `audsp` (audio spooler) binary to handle audio output. If this binary doesn't have execute permissions, you'll get "pipe_open: failed to start audsp" errors.

Build the DEB Package

```bash
cd ~/app_installs/festival/196/build

# Build the package
sudo dpkg-deb --build festival-1.96-deb

# Rename to descriptive name
ARCH=$(dpkg --print-architecture)
sudo mv festival-1.96-deb.deb festival_1.96-nitech1_${ARCH}.deb

# Change ownership back to your user
sudo chown $USER:$USER festival_1.96-nitech1_${ARCH}.deb
```

Verify the Package

Check package contents and metadata:

```bash
# List contents
dpkg -c festival_1.96-nitech1_amd64.deb

# Check package info
dpkg -I festival_1.96-nitech1_amd64.deb

# Verify audsp has execute permissions in the package
dpkg -c festival_1.96-nitech1_amd64.deb | grep audsp
```

The audsp entry should show `rwxr-xr-x` (executable) permissions.

Step 11: Install and Test the Package

Install the Package

```bash
sudo dpkg -i festival_1.96-nitech1_amd64.deb
```

Test All Functionality

Test the different ways to use Festival:

```bash
# Test 1: Command-line TTS (--tts mode)
echo 'Hello world' | festival --tts

# Test 2: text2wave with aplay
echo "Hello world" | text2wave | aplay

# Test 3: Direct Scheme command
echo '(SayText "Hello world")' | festival

# Test 4: Create WAV file
echo "Hello world" | text2wave -o test.wav
aplay test.wav
```

Test Interactive Mode

```bash
festival
```

In interactive mode:

```scheme
; List all available voices
festival> (voice.list)

; Test the default SLT voice (female)
festival> (SayText "This is the SLT female voice")

; Try other voices
festival> (voice_nitech_us_awb_arctic_hts)
festival> (SayText "This is the AWB male voice")

festival> (voice_nitech_us_clb_arctic_hts)
festival> (SayText "This is the CLB female voice")

; Exit
festival> (quit)
```

Verify Installation Paths

Start Festival and verify it's using the correct paths:

```bash
festival
```

```scheme
festival> (car load-path)
festival> (Parameter.get 'Audio_Method)
festival> (Parameter.get 'Audio_Command)
```

Expected output:

- `load-path` should show `/usr/local/share/festival/...`
- `Audio_Method` should be `Audio_Command`
- `Audio_Command` should be `"aplay -q -c 1 -t raw -f s16 -r $SR $FILE"`

Create Personal Configuration (Optional)

To customize Festival for your user, create `~/.festivalrc`:

```bash
cat > ~/.festivalrc << 'EOF'
;;; Personal Festival configuration

; Use ALSA for audio
(Parameter.set 'Audio_Method 'Audio_Command)
(Parameter.set 'Audio_Command "aplay -q -c 1 -t raw -f s16 -r $SR $FILE")

; Pitch adjustment: Speed up by 5% to raise pitch slightly
;(Parameter.set 'Audio_Command "aplay -q -c 1 -t raw -f s16 -r $(($SR*105/100)) $FILE")

; Set default voice
(set! voice_default 'voice_nitech_us_slt_arctic_hts)

; Volume boost (doubles amplitude)
(set! default_after_synth_hooks
   (list (lambda (utt) (utt.wave.rescale utt 2.0 t))))
EOF
```

Automated Build Script

For convenience, an automated bash script is available that performs all these steps automatically. Save this as `build_festival_196_nitech.sh`:

The script is available in the artifacts and includes:

- Automatic dependency checking
- All compilation steps with proper flags
- Correct FTLIBDIR configuration
- Proper permission setting for audsp
- Complete package creation
- Error handling and progress reporting

To use:

```bash
chmod +x build_festival_196_nitech.sh
./build_festival_196_nitech.sh
```

The script creates a `festival196nitech` directory and builds everything automatically, producing a ready-to-install `.deb` package in about 10-15 minutes.

Troubleshooting

Issue: "malloc(): corrupted top size" error

Cause: Missing the `-D_GLIBCXX_USE_CXX11_ABI=0` flag during compilation.

Solution: This flag is absolutely critical. Ensure you compiled both Speech Tools and Festival with this flag. If not, go back to Steps 4 and 7 and recompile.

Issue: "can't open /dev/dsp" error

Cause: OSS audio system not available on modern Linux.

Solution: This is normal in interactive mode before configuration is loaded. The package includes ALSA configuration in `/usr/local/share/festival/siteinit.scm` which handles this automatically. If you still get this error after installation, verify:

```bash
cat /usr/local/share/festival/siteinit.scm
```

Issue: "pipe_open: failed to start audsp" or "Audio spooler has died unexpectedly"

Cause: The `audsp` binary doesn't have execute permissions.

Solution:

```bash
sudo find /usr/local/share/festival -type f -name "audsp" -exec chmod 755 {} \;
```

To verify audsp is executable:

```bash
find /usr/local/share/festival -name "audsp" -exec ls -la {} \;
```

Should show `rwxr-xr-x` permissions.

Issue: Festival looks in wrong directory for libraries

Cause: `FTLIBDIR` wasn't set correctly in `config/project.mak` before compilation.

Solution: Verify Step 6 was completed correctly:

```bash
cat festival/config/project.mak | grep FTLIBDIR
```

Should show: `FTLIBDIR = /usr/local/share/festival`

If not, you must recompile Festival after fixing this.

Issue: `--tts` mode doesn't work but interactive mode does

Cause: The `audsp` binary lacks execute permissions.

Solution: See "pipe_open: failed to start audsp" above.

Issue: Voice not found or "unbound variable" errors

Cause: Voices not properly installed or HTS engine support missing.

Solution: Verify files exist:

```bash
ls -la /usr/local/share/festival/voices/us/nitech_us_slt_arctic_hts/
ls -la /usr/local/share/festival/hts.scm
```

If missing, reinstall the package or copy the voices manually from the build directory.

Issue: Multiple definition linker errors during compilation

Cause: Missing `-Wl,--allow-multiple-definition` flags.

Solution: Ensure you modified `config/compilers/gcc_defaults.mak` in both Speech Tools and Festival (Steps 3 and 5).

Uninstalling

To remove the package:

```bash
sudo dpkg -r festival
```

To remove including configuration files:

```bash
sudo dpkg --purge festival
```

Directory Structure

After successful installation, the directory structure is:

```
/usr/local/
├── bin/
│   ├── festival            # Main Festival executable
│   ├── festival_client      # Client for Festival server
│   ├── text2wave            # Convert text to WAV files
│   ├── ch_wave              # Speech Tools wave manipulation
│   ├── ch_track             # Speech Tools track manipulation
│   └── wagon                # CART tree builder
├── share/
│   └── festival/
│       ├── hts.scm          # HTS engine support
│       ├── init.scm         # Festival initialization
│       ├── siteinit.scm     # Site configuration (ALSA audio, default voice)
│       ├── etc/
│       │   └── */audsp      # Audio spooler binary (must be executable!)
│       ├── voices/
│       │   └── us/
│       │       ├── nitech_us_slt_arctic_hts/  # Female voice (default)
│       │       ├── nitech_us_awb_arctic_hts/  # Male voice
│       │       ├── nitech_us_clb_arctic_hts/  # Female voice
│       │       └── [other voices...]
│       ├── dicts/           # Lexicons
│       └── [other lib files]
└── share/doc/festival/      # Documentation

/etc/festival/
└── siteinit.scm             # Reference copy of configuration
```

Performance Notes

This guide uses `-O0` (no optimization) for maximum compatibility. If you need better performance, you can try `-O2`:

```bash
make CXXFLAGS="-O2 -Wall -fpermissive -std=c++98 -D_GLIBCXX_USE_CXX11_ABI=0 -DSUPPO
```

However, test thoroughly as higher optimization may expose hidden bugs in the old codebase.

Why Festival 1.96?

Festival 1.96 is specifically required for compatibility with the Nitech HTS voices. Newer Festival versions (2.x) are not backward compatible with these high-quality voices.

Available Voices in This Package

The Nitech HTS voices included are:

- **nitech_us_slt_arctic_hts** - Female (most popular, default)
- **nitech_us_awb_arctic_hts** - Male
- **nitech_us_bdl_arctic_hts** - Male
- **nitech_us_clb_arctic_hts** - Female
- **nitech_us_rms_arctic_hts** - Male
- **nitech_us_jmk_arctic_hts** - Male

These are significantly higher quality than the default Festival voices and are well-suited for applications requiring natural-sounding speech synthesis.

Summary of Critical Points

1. **Compiler flags are essential:** `-D_GLIBCXX_USE_CXX11_ABI=0` is non-negotiable for modern systems
2. **Linker flags required:** Add `-Wl,--allow-multiple-definition` to both projects
3. **Path configuration is critical:** Modify `FTLIBDIR` in `config/project.mak` before compiling
4. **Compile order matters:** Always compile Speech Tools before Festival
5. **Include Nitech voices:** They provide the best quality for this Festival version
6. **Test before packaging:** Verify Festival works from build directory first
7. **Set proper permissions:** Everything in the DEB must be owned by root
8. **audsp must be executable:** This is critical for `--tts` mode to work
9. **siteinit.scm location matters:** Must be in `/usr/local/share/festival/` not just `/etc/festival/`

Distribution

Your final `festival_1.96-nitech1_amd64.deb` package can be:

- Installed on any compatible Debian/Ubuntu system
- Shared with others
- Uploaded to a personal repository
- Distributed via file hosting

The package is completely self-contained and includes everything needed for high-quality text-to-speech synthesis.

Testing Checklist

After installation, verify:

- ✓ `echo 'test' | festival --tts` works (tests audsp permissions)
- ✓ `echo 'test' | text2wave | aplay` works (tests audio pipeline)
- ✓ Interactive mode works: `festival` then `(SayText "test")`
- ✓ Path is correct: `(car load-path)` shows `/usr/local/share/festival/...`
- ✓ Audio configured: `(Parameter.get 'Audio_Method)` shows `Audio_Command`
- ✓ Voices available: `(voice.list)` shows nitech voices
- ✓ Can switch voices: `(voice_nitech_us_awb_arctic_hts)` works

References

- Festival Homepage: http://www.cstr.ed.ac.uk/projects/festival/
- Festival Manual: http://www.cstr.ed.ac.uk/projects/festival/manual/
- Festival 1.96 Downloads: http://festvox.org/packed/festival/1.96/
- Nitech HTS Homepage: http://hts.sp.nitech.ac.jp/
- Mirrored Nitech Voices: http://erewhon.superkuh.com/nitech_voices_for_festival_196.tar.gz

Conclusion

Following this guide results in a fully functional Festival 1.96 installation packaged as a distributable Debian package with high-quality Nitech HTS voices. The entire process takes about 30-45 minutes depending on compilation speed.

The resulting package provides professional-quality text-to-speech synthesis that works on any modern Debian-based Linux distribution, with proper ALSA audio support and the best available voices for Festival 1.96.