🚀 Minishell Project - Complete Documentation

Bu proje, bash shell'inin temel özelliklerini implementasyonu olan **minishell** projesidir. 42 School standartlarına uygun olarak geliştirilmiştir.

📁 Dosya Yapısı

```
Minishell/
├── 📄 MINISHELL_DOCUMENTATION.md    # Ana dokümantasyon
├── 📄 TECHNICAL_ANALYSIS.md         # Teknik analiz
├── 📄 TEST_SCENARIOS.md            # Test senaryoları
├── 📄 QUICK_START_GUIDE.md         # Bu dosya
├── 📄 minishell.c                  # Ana program
├── 📄 minishell.h                  # Header dosyası
├── 📄 Makefile                     # Build konfigürasyonu
└── src/                           # Kaynak kodlar
    ├── tokenizer/                 # Tokenization modülü
    ├── parser/                    # Parsing modülü
    ├── executor/                  # Execution modülü
    ├── expander/                  # Variable expansion
    ├── builtin/                   # Builtin komutlar
    └── redirect/                  # Redirection handling
```

🎯 Özellikler

☑️ Desteklenen Özellikler

- **Komut Çalıştırma**: External ve builtin komutlar
- **Pipeline**: `|` operatörü ile komut zinciri
- **Redirection**: `<`, `>`, `>>` operatörleri
- **Heredoc**: `<<` ile çoklu satır giriş
- **Variable Expansion**: `$VAR` ve `$?` desteği
- **Quote Handling**: `'single'` ve `"double"` quotes
- **Signal Handling**: Ctrl+C desteği
- **Builtin Commands**: echo, cd, pwd, export, unset, env, exit

🔧 Builtin Komutlar

```
echo [-n] [text...]        # Metin yazdırma
cd [directory]             # Dizin değiştirme
pwd                        # Mevcut dizin
export [VAR=value]         # Environment variable tanımlama
unset [VAR]               # Environment variable silme
env                       # Environment variable'ları listeleme
exit [code]               # Shell'den çıkış
```

🚀 Kurulum ve Çalıştırma

Gereksinimler

- GCC compiler
- readline library
- Make utility

Derleme

```
# Projeyi derle
make

# Temizlik
make clean      # Object dosyalarını sil
make fclean     # Tüm oluşturulan dosyaları sil
make re         # Temizle ve yeniden derle
```

Çalıştırma

```
# Interactive mode
./minishell

# Script mode (gelecek implementasyon)
./minishell script.sh
```

📖 Kullanım Örnekleri

◇ Temel Komutlar

```
guest@minishell $ echo "Hello World"
Hello World

guest@minishell $ pwd
/current/directory

guest@minishell $ ls -la
# External command execution
```

◇ Pipeline Kullanımı

```
guest@minishell $ env | sort | head -5
# Environment variables, sorted, first 5
```

```
guest@minishell $ cat file.txt | grep pattern | wc -l
# Count matching lines
```

◇ Redirection Örnekleri

```
guest@minishell $ echo "data" > output.txt
guest@minishell $ cat < input.txt > output.txt
guest@minishell $ command 2>> error.log
```

◇ Heredoc Kullanımı

```
guest@minishell $ cat << EOF
> Line 1
> Line 2
> EOF
Line 1
Line 2
```

◇ Variable Operations

```
guest@minishell $ export NAME="John"
guest@minishell $ echo "Hello $NAME"
Hello John

guest@minishell $ echo $?
0  # Exit status of last command
```

🔧 Teknik Detaylar

Mimari

```
Input → Tokenizer → Expander → Parser → Executor → Output
```

Memory Management

- RAII pattern kullanımı
- Automatic cleanup
- Zero memory leaks
- should_exit flag pattern

Process Management

- fork/exec pattern
- Proper signal handling
- Pipeline process chaining
- Exit status propagation

🧪 Test Etme

Manuel Test

```
# Temel testler
echo hello | grep hello
cat /etc/passwd | head -5
export TEST=value && echo $TEST

# Error handling
nonexistent_command
echo hello |
```

Memory Leak Kontrolü

```
valgrind --leak-check=full ./minishell
# Expected: "All heap blocks were freed -- no leaks are possible"
```

Norm Kontrolü

```
norminette src/ includes/
# Expected: No errors
```

📋 Çözülen Problemler

🟢 Memory Leaks

- **Problem**: Direct exit() calls causing memory leaks
- **Çözüm**: should_exit flag mechanism with proper cleanup

🟢 Exit Code Compatibility

- **Problem**: Non-bash-compatible exit codes
- **Çözüm**: 8-bit masking (& 255) for bash compatibility

🟢 Signal Handling

- **Problem**: Heredoc requiring double Ctrl+C
- **Çözüm**: Direct _exit(130) on readline NULL return

42 Norm Compliance

- **Problem**: Global variables usage
- **Çözüm**: Struct-based state management

Hata Kodları

```
0   - Success
1   - General error
2   - Syntax error
126 - Permission denied
127 - Command not found
130 - Interrupted by signal (Ctrl+C)
```

Debugging

Debug Mode

```
# Compile with debug flags
make debug

# GDB debugging
gdb ./minishell
(gdb) run
(gdb) bt  # Backtrace on crash
```

Log Output

```
# Enable debug output (if implemented)
export MINISHELL_DEBUG=1
./minishell
```

Referanslar

Dokümantasyon

- `MINISHELL_DOCUMENTATION.md` - Kapsamlı kullanım kılavuzu
- `TECHNICAL_ANALYSIS.md` - Teknik implementasyon detayları
- `TEST_SCENARIOS.md` - Test senaryoları ve örnekler

External References

- Bash Manual
- Advanced Programming in UNIX Environment
- 42 Norm Guidelines

🤝 Katkıda Bulunma

Code Style

- 42 Norm kurallarına uygunluk
- Fonksiyon başına max 25 satır
- Global variable yasağı
- Proper error handling

Commit Messages

```
feat: implement heredoc functionality
fix: resolve memory leak in tokenizer
docs: update technical documentation
test: add pipeline stress tests
```

📞 İletişim

- **Yazarlar**: haloztur, musoysal
- **Proje**: 42 School Minishell
- **Tarih**: Temmuz 2025
- **Versiyon**: 1.0

🎯 Hızlı Başlangıç Komutları

```bash
# Clone and build
git clone [repo] && cd minishell
make

# Test basic functionality
./minishell
guest@minishell $ echo "Hello, $(whoami)!"
guest@minishell $ ls | head -3
guest@minishell $ exit

# Memory check
make && valgrind --leak-check=full ./minishell
```

Bu proje 42 School Minishell project requirements doğrultusunda geliştirilmiştir. Tüm implementasyon bash-compatible ve norm-compliant'tır.