

(This file is a duplicate of uClinux-dist/Documentation/Adding-User-Apps-HOWTO.)

## Adding User Applications to the uClinux Distribution

---

D. P. Siddons  
9th Dec. 2001

This document gives simple instructions for adding a user-written application to the uClinux configuration system. Entries must be added to three files, and an appropriate Makefile must exist in the user application source directory, which must be put in user (all directory names here are given relative to the uClinux top directory. In my system this is /home/peter/uClinux-dist).

Files to edit:

user/Makefile

Add a line to the file like

```
dir_$(CONFIG_USER_FOO_FOO)          += foo
```

This adds the directory 'foo' to the list of directories to be built. I added mine in alphabetical order. The order doesn't seem to matter.

config/Configure.help

This file contains the text which is presented on request during the config.

Add a block like

```
CONFIG_USER_FOO_FOO
```

This program does fooley things to your bars.

The text must be indented two spaces, and there must be no empty lines. Lines should be <70 chars long.

config/config.in:

Add a line in the appropriate menu section (i.e. in the program group you want your app to show up in during 'make config'; I used 'misc'), like

```
bool 'foo'          CONFIG_USER_FOO_FOO
```

The repetition of FOO allows for directories which contain multiple executables. Thus, if the user directory 'foo' contained code to make 'foo' and 'bar', each gets its own config line if an additional entry is made like

```
bool 'bar'          CONFIG_USER_FOO_BAR
```

Next, there needs to be a proper /user/foo/Makefile. The Makefile should follow the following template:

---

```
EXEC = foo  
OBSJ = foo.o
```

```

all: $(EXEC)

$(EXEC): $(OBJS)
    $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

romfs:
    $(ROMFSINST)    /bin/$(EXEC)

clean:
    -rm -f $(EXEC) *.elf *.gdb *.o
-----

```

If more than one executable is built in the foo directory, as above, then the Makefile should look like

```

-----
EXECS = foo bar
OBJS = foo.o bar.o

all: $(EXECS)

$(EXECS): $(OBJS)
    $(CC) $(LDFLAGS) -o $@ $@.o $(LDLIBS)

romfs:
    $(ROMFSINST) -e CONFIG_USER_FOO_FOO    /bin/foo
    $(ROMFSINST) -e CONFIG_USER_FOO_BAR    /bin/bar
-----

```

More complex makefiles are of course possible. The reader is encouraged to browse the user tree for examples.

When all this is set up, doing the standard 'make xconfig; make dep; make' should build the app and install it in romfs and hence in the target system image.bin.