Wstęp do programowania w języku C

Automatyczna kompilacja - Makefile
Prosty komunikator z użyciem GTK+ 3.0 i ATD

Marek Piotrów - Wykład 12

13 stycznia 2022

Automatyzacja kompilacji projektów - Makefile

- Standardowy program `make` automatyzuje kompilację projektów.
- `make` czyta skrypt(y) `Makefile` i na ich podstawie ustala, które pliki wymagają kompilacji.
- Następnie kompiluje je według podanych w `Makefile` reguł.
- Po wprowadzeniu zmian do projektu kompilowane są tylko moduły, na które zmiana mogła mieć wpływ.
- Istnieje kilka wersji tego programu: np. GNU make, czy Miscrosoft make.
- `Makefile` zawiera zbiór reguł dla `make` opisujący zależności między plikami i reguły kompilacji.
- Narzędzie `cbp2make` automatycznie tworzy `Makefile` dla projektu Code::Blocks.

Trywialny Makefile

```
szesciokat : punkty.c szesciokat.c punkty.h
	gcc -std=c11 -Wall -Wextra -Werror punkty.c szesciokat.c -o szesciokat -lm
```

Najprostszy Makefile

```
# najprostszy makefile dla programu szesciokat

szesciokat: punkty.o szesciokat.o
	gcc -std=c11 -Wall -Wextra -Werror punkty.o szesciokat.o -o szesciokat -lm

punkty.o: punkty.c punkty.h
	gcc -std=c11 -Wall -Wextra -Werror -c punkty.c -o punkty.o

szesciokat.o: szesciokat.c punkty.h
	gcc -std=c11 -Wall -Wextra -Werror -c szesciokat.c -o szesciokat.o
```

Prosty Makefile

```
# prosty makefile dla programu: szesciokat

CC=gcc
CFLAGS= -std=c11 -Wall -Wextra -Werror
# ogolne flagi kompilacji dla modulow

LDFLAGS= -lm
# ogolne flagi konsolidacji programu

DEPS = punkty.h
NAME = szesciokat

$(NAME): punkty.o szesciokat.o
	$(CC) $(CFLAGS) punkty.o szesciokat.o -o $(NAME) $(LDFLAGS)

punkty.o: punkty.c $(DEPS)
	$(CC) -c $(CFLAGS) punkty.c -o punkty.o

szesciokat.o: szesciokat.c $(DEPS)
	$(CC) -c $(CFLAGS) szesciokat.c -o szesciokat.o
```

Standardowy Makefile

```
# standardowy makefile dla programu: szesciokat

CC=gcc
CFLAGS = -std=c11 -Wall -Wextra -Werror
LFLAGS = -lm

NAME = szesciokat
# nazwa programu wynikowego
SRC =  szesciokat.c punkty.c
DEPS = punkty.h
# nazwy wszystkich plikow zrodlowych
OBJS = szesciokat.o punkty.o
# nazwy wszystkich modulow

YOU : $(SRC) $(NAME)

$(NAME): $(OBJS)
	$(CC) $(CFLAGS) $(OBJS) -o $(NAME) $(LFLAGS)

%.o: %.c $(DEPS)
	$(CC) $(CFLAGS) -c -o $@ $<

clean:
	rm -f $(OBJS) $(NAME)
```

Makefile dla pracy magisterskiej I

```
OPTIONS = -DNDEBUG
# OPTIONS = -DPOOR_LIBS -DNDEBUG
# Use the above line if you have problems with compilation due to
# missing declarations. This enables including of "fixes.h", where you
# can place or include what is missing.
# when finished, uncomment -DNDEBUG to remove debugging code

CFLAGS = -s -funsigned-char -O2 # -fno-exceptions
# when finished, add -O2 to CFLAGS
# -fno-exceptions is for gcc 2.8.x
# (I don't use exceptions -> executable will be smaller)
LFLAGS = -lm # -lstdc++

SRC = euphoria.l euphoria.y symtab.cc symtab.h fixes.h init.cc code.h code.cc \
errors.h seq.cc seq.h
OBJS = symtab.o init.o code.o euphoria.o seq.o
NAME = euphoria

YOU : $(SRC) $(NAME)

$(NAME) : $(OBJS)
	gcc $(CFLAGS) $(OPTIONS) -o $(NAME) $(OBJS) $(LFLAGS)

lex.yy.c : euphoria.l
	flex -s euphoria.l

euphoria.tab.c : euphoria.y
```

Makefile dla pracy magisterskiej II

```
	bison euphoria.y

euphoria.o : euphoria.tab.c lex.yy.c
	gcc -c $(CFLAGS) $(OPTIONS) -x c++ $< -o $@

%.o : %.cc
	gcc -c $(CFLAGS) $(OPTIONS) $< -o $@
```

Komunikator

ZADANIE: Napisać prosty komunikator z okienkowym interfejsem, który:

- Po uruchomieniu dwóch kopii A i B tego programu przesyła teksty pomiędzy kopiami używając systemowych kolejek fifo. Litera A lub B powinna być pierwszym argumentem wywołania programu.
- Uwzględnia różne implementacje kolejek fifo w systemach windows i linux.
- Używa modułów i abstrakcyjnych typów danych do ukrycia różnic i szczegółów implementacji.
- Sygnalizuje błędy komunikacji i sposobu wywołania programu.

Komunikator

ROZWIĄZANIE:

- Użyć biblioteki GTK+ do zdefiniowania i obsługi interfejsu okienkowego. Będzie to pierwszy moduł rozwiązania.
- Napisać oddzielne moduły implementujące ATD dla systemów linux i windows.
- Do sygnalizacji błędów drugi moduł użyje funkcji: `void pokazBlad(char *blad);` z pierwszego modułu.

Interfejs modułu komunikacyjnego: fifo.h

```
#include <stdbool.h>

typedef struct pipes *PipesPtr;

PipesPtr initPipes(int argc, char *argv[]);
void     sendStringToPipe(PipesPtr channel, const char *data);
bool     getStringFromPipe(PipesPtr channel, char *buffer, size_t size);
void     closePipes(PipesPtr channel);
```

Interfejs komunikatora w GTK (gtk-talk.c) I

```c
#include <string.h>
#include <math.h>
#include <gtk/gtk.h>
#include "fifo.h"
// kompilacja pod linuxem z lin-fifo.c a pod windowsem z win-fifo.c
// linux:   gcc -std=c11 -Wall -o gtk-talk gtk-talk.c lin-fifo.c `pkg-config gtk+-3.0 --cflags --libs`
// windows: gcc -std=c11 -Wall -o gtk-talk gtk-talk.c win-fifo.c `pkg-config gtk+-3.0 --cflags --libs`

#define MAKS_DL_TEKSTU 10000

static GtkWidget *window, *bufor;
static PipesPtr potoki;
static char *moj_id, *twoj_id;

static void przekaz_tekst( GtkWidget *widget,GtkWidget *text);
static gboolean pobierz_tekst(gpointer data);
static void zakoncz(GtkWidget *widget, gpointer data);

void pokazBlad(char *komunikat)
{
   GtkWidget *dialog;
   dialog=gtk_message_dialog_new (GTK_WINDOW(window),GTK_DIALOG_DESTROY_WITH_PARENT,
                                  GTK_MESSAGE_ERROR,GTK_BUTTONS_CLOSE,"%s",komunikat);
   gtk_dialog_run (GTK_DIALOG (dialog));
   gtk_widget_destroy (dialog);
}
```

Interfejs komunikatora w GTK (gtk-talk.c) II

```
int main(int argc,char *argv[])
{
   if ((potoki=initPipes(argc,argv)) == NULL)
      return 1;
   if (argc == 2 && strcmp(argv[1],"A") == 0) { twoj_id="B > "; moj_id="A > ";}
   else { moj_id="B > "; twoj_id="A > ";}

   gtk_init(&argc, &argv);

   gchar naglowek[30];
   sprintf(naglowek,"Hej %c, tu %c, porozmawiajmy :-)",twoj_id[0],moj_id[0]);
   window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(window),naglowek);
   g_signal_connect(G_OBJECT(window), "destroy",G_CALLBACK(zakoncz), NULL);
   gtk_container_set_border_width(GTK_CONTAINER(window), 10);

   GtkWidget *grid = gtk_grid_new();
   gtk_grid_set_row_spacing(GTK_GRID(grid), 1);
   gtk_grid_set_row_homogeneous(GTK_GRID(grid), TRUE);
   gtk_grid_set_column_homogeneous(GTK_GRID(grid), TRUE);
   gtk_container_add(GTK_CONTAINER(window), grid);

   bufor = (GtkWidget *)gtk_text_buffer_new (NULL);
   GtkWidget *text_view = gtk_text_view_new_with_buffer (GTK_TEXT_BUFFER(bufor));
   gtk_text_view_set_wrap_mode (GTK_TEXT_VIEW(text_view), GTK_WRAP_WORD);
   gtk_text_view_set_editable(GTK_TEXT_VIEW(text_view), FALSE);
```

Interfejs komunikatora w GTK (gtk-talk.c) III

```
    gtk_text_view_set_cursor_visible(GTK_TEXT_VIEW(text_view), FALSE);

    GtkWidget *scrolled_window = gtk_scrolled_window_new (NULL, NULL);
    gtk_scrolled_window_set_policy (GTK_SCROLLED_WINDOW(scrolled_window),
GTK_POLICY_AUTOMATIC,
                                    GTK_POLICY_AUTOMATIC);
    gtk_container_add (GTK_CONTAINER (scrolled_window), text_view);
    gtk_container_set_border_width (GTK_CONTAINER(scrolled_window), 1);
    gtk_grid_attach(GTK_GRID(grid), scrolled_window, 0, 0, 60, 20);

    GtkWidget *text = gtk_entry_new();
    gtk_entry_set_max_length(GTK_ENTRY(text), MAKS_DL_TEKSTU);
    gtk_entry_set_text(GTK_ENTRY(text), "");
    g_signal_connect(G_OBJECT(text), "activate",G_CALLBACK(przekaz_tekst),(gpointer) text);
    gtk_grid_attach(GTK_GRID(grid), text, 0, 20, 60, 1);

    GtkWidget *button=gtk_button_new_with_label("koniec");
    g_signal_connect(G_OBJECT(button), "clicked",G_CALLBACK(zakoncz), NULL);
    gtk_grid_attach(GTK_GRID(grid), button, 25, 21, 10, 1);

    g_timeout_add(100,pobierz_tekst,NULL);

    gtk_widget_show_all(window);
    gtk_widget_grab_focus(text);
    gtk_main();
    return 0;
```

Interfejs komunikatora w GTK (gtk-talk.c) IV

```
}

static void przekaz_tekst( GtkWidget *widget,GtkWidget *text)
{
   gchar wejscie[MAKS_DL_TEKSTU+5];

   sendStringToPipe(potoki, gtk_entry_get_text (GTK_ENTRY (text)));

   strcpy(wejscie,moj_id);
   strcpy(wejscie+strlen(wejscie),gtk_entry_get_text (GTK_ENTRY (text)));
   strcat(wejscie,"\n");

   gtk_text_buffer_insert_at_cursor (GTK_TEXT_BUFFER(bufor),wejscie,-1);
   gtk_entry_set_text(GTK_ENTRY(text), "");

}

static gboolean pobierz_tekst(gpointer data)
{
  gchar wejscie[MAKS_DL_TEKSTU+5];

  strcpy(wejscie,twoj_id);
  if (getStringFromPipe(potoki,wejscie+strlen(wejscie),MAKS_DL_TEKSTU)) {
     strcat(wejscie,"\n");
     gtk_text_buffer_insert_at_cursor (GTK_TEXT_BUFFER(bufor),wejscie,-1);
  }
```

Interfejs komunikatora w GTK (gtk-talk.c) V

```
  return TRUE;
}

static void zakoncz(GtkWidget *widget, gpointer data)
{
  closePipes(potoki);
  gtk_main_quit();
}
```