PODSTAWY SYSTEMÓW OPERACYJNYCH
laboratorium nr 6 (zadania do lab 5)

Kompilacja programu z poziomu konsoli - Linux
Obsługa systemu plików.

PREREKWIZYTY:
Należy zalogować się na root-a i:
-dokonać aktualizacji informacji o bazie dostępnego oprogramowania
    apt-get update
-dokonać instalacji gcc
    apt-get install gcc

TESTOWANIE ŚRODOWISKA:
W celu przetestowania należy skompilować i uruchomić pierwszy program:

```
#include <stdio.h>


int main(void)
{
    puts ("Witam!");

return 0;
}
```

Za pomocą edytora Nano lub innego zapisz powyższy program w pliku (nazwa.c)
Skompiluj program:
    gcc nazwa_pliku

Po wykonaniu komilacji w katalogu powinien pojawić się plik a.out.
Jest to plik wykonywalny z programem po uruchomieniu powinien wyświetlić komunikat "Witam!"

Kompilacja gcc plik.c automatycznie tworzy plik binarny o nazwie a.out.
Jeżeli chcemy aby plik wyjściowy miał konkretną nazwę używamy opcji -o.
gcc -o nazwa kod.c

Jeśli chcemy podejrzeć jak wygląda nasz porogram w assemblerze to możemy wydać polecenie:
    gcc -S nazwa_pliku
Zostanie utworzony plik z końcówką ".s"

Aby uruchomić plik wykonywalny należy go poprzedzić "./" (kropka i ukośnik wznoszący)

ZADANIA DO WYKONANIA:
1. Program sprawdza czy jest możliwy dostęp do pliku - jako parametr przyjmuje nazwę pliku

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char** argv)
{
    if (argc<2)
    {
        printf("Uzycie: %s plik \n", argv[0]);
        exit(1);
    }

    if(access(argv[1], R_OK) == -1)
        fprintf(stderr, "Nie mozesz czytac z pliku %s \n", argv[1]);

    else    printf("Czytanie z pliku dozwolone \n");

    if(access(argv[1], W_OK) == -1)
        fprintf(stderr, "Nie mozesz pisać do pliku %s \n", argv[1]);

    else     printf("Pisanie do pliku dozwolone \n");

    if(access(argv[1], X_OK) == -1)
```

```
71          fprintf(stderr, "Nie mozesz uruchamiać pliku %s \n", argv[1]);
72
73      else     printf("Uruchamianie pliku dozwolone \n");
74
75      exit(0);
76  }
77  ##########################################################################
78
79  2.Poniższy program pokazuje parametry środowiska oraz argumenty wpisane w linii
    poleceń:
80  ##########################################################################
81  #include <stdio.h>
82  #include <stdlib.h>
83
84  int i=0;
85  int main(int argc, char* argv[], char* env[])
86  {
87      printf("Parametry srodowiska Linux: \n");
88      do {
89      printf("%s \n", env[i]);
90      i++;
91      }
92      while (env[i]!=NULL);
93          printf("Lista parametrow programu: \n");
94
95     for(i=1;i<=argc-1;i++) printf("%s \n", argv[i]);
96
97      printf("Nazwa programu: \n");
98      printf("%s \n", argv[0]);
99      return 0;
100 }
101 ##########################################################################
102
103 3. program pokazuje strukture stat dla podanego pliku
104 ##########################################################################
105 #include <stdlib.h>
106 #include <stdio.h>
107 #include <sys/stat.h>
108
109 int main(int argc, char** argv)
110 {
111     struct stat statystyka;
112
113     if(argc<2)
114     {
115     printf("Uzycie: %s plik \n",argv[0]);
116     exit(1);
117     }
118     if(stat(argv[1], &statystyka) == -1)
119     {
120         perror("");
121         return(-1);
122     }
123     printf("Dane dla %s odczytane za pomoca funkcji 'stat' :\n", argv[1]);
124     printf("Urzadzenie logiczne (st_dev)   : %d\n", statystyka.st_dev);
125     printf("Numer i-wezla (st_ino)   : %d\n", statystyka.st_ino);
126     printf("Uprawnienia (st_mode)  : %d\n", statystyka.st_mode);
127     printf("Liczba laczy (st_nlink) : %d\n", statystyka.st_nlink);
128     printf(" UID (st_uid) : %d\n", statystyka.st_uid);
129     printf(" GID (st_gid) : %d\n", statystyka.st_gid);
130     printf("Opis urzadzenia (st_rdev) : %d\n", statystyka.st_rdev);
131     printf("Wielkosc pliku (st_size) : %ld\n", statystyka.st_size);
132     printf("Ostatni odczyt (st_atime) : %d\n", statystyka.st_atime);
133     printf("Ostatnia modyfikacja (st_mtime) : %d\n", statystyka.st_mtime);
134     printf("Ostatnia zmiana w 'stat' (st_ctime) : %d\n", statystyka.st_ctime);
135     printf("Wielkosc bloku I/O (st_blksize) : %ld\n", statystyka.st_blksize);
136     printf("Ilosc fiz. blokow dla pliku (st_blocks) : %ld\n",
        statystyka.st_blocks);
137 exit(0);
138 }
139 ##########################################################################
140
141 4. program do kasowania pliku
```

```
142 ##############################################################################
143 #include <unistd.h>
144 #include <stdio.h>
145 #include <stdlib.h>
146
147 FILE *p;
148 char *uzycie = "Uzycie: del sciezka_do_pliku \n";
149 char c[0];
150
151 int usun(const char *plik)
152 {
153     /*sprawdzanie za pomoca funkcji open czy plik istnieje*/
154     if ((p = fopen(plik,"r")) == NULL)
155     {
156         fprintf(stderr, "Plik %s nie istnieje \n",plik);
157         return(-1);
158     }
159
160     fclose(p);
161
162     printf("Czy mam na pewno usunac %s? T/N\n", plik);
163     scanf("%s", c);
164
165     if(c[0] == 't' || c[0] == 'T')
166     {
167         remove(plik);
168         printf("Plik usuniety\n");
169         return(0);
170     }
171
172     printf("Anulowano...\n");
173
174 }
175
176 int main(int argc, char** argv)
177 {
178     if (argc<2)
179     {
180         fprintf(stderr, uzycie);
181         exit(1);
182     }
183
184     usun(argv[1]);
185     exit(0);
186 }
187 ##############################################################################
188
189
190 5. program wykorzystauje procedury biblioteczne
191 program kopiuje plik1 do plik2
192 ##############################################################################
193 */
194
195 #include <stdio.h>
196 #include <stdlib.h>
197
198 /*funkcja kopiujaca*/
199 int copyfile(const char *plik1, const char *plik2)
200 {
201     FILE *in, *out;
202     int c;
203
204     if ((in = fopen(plik1,"r")) == NULL)
205     return(-1);
206
207     if ((out = fopen(plik2,"w")) == NULL)
208     {
209     fclose(in);
210     return(-2);
211     }
212
213     while((c = getc(in)) != EOF)
214     putc(c, out);
```

```
215
216         fclose(in);
217         fclose(out);
218     }
219     ////////////////////////////////
220      int main(int argc, char** argv)
221      {
222          if(argc<3)
223          {
224              printf("Uzyj: %s plik_zrodlo plik_cel\n",argv[0]);
225          exit(1);
226          }
227
228          copyfile(argv[1],argv[2]);
229      }
230     ##############################################################################
231
232     Źródła:
233     http://wazniak.mimuw.edu.pl/
234     https://4programmers.net/C/Artykuły/Jak_programować_w_Linuksie
235
```