PODSTAWY SYSTEMÓW OPERACYJNYCH
laboratorium nr 5b

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"

Przykłady zastosowania operacji plikowych.
Listing 1 przedstawia program do kopiowania pliku.
W programie wykorzystano funkcje systemowe open, creat, read, write i close.
Nazwy plików przykazywane są jako argumenty w linii poleceń przy uruchamianiu programu.
Jako pierwszy argument przekazywana jest nazwa istniejącego pliku źródłowego,
a jako drugi argument przekazywana jest nazwa pliku docelowego,
który może zostać dopiero utworzony:

Listing 1: Kopiowanie pliku:

```
   #include <fcntl.h>
   #include <stdio.h>
   #include <stdlib.h>
3  #define MAX 512

   int main(int argc, char* argv[]){
6   char buf[MAX];
    int desc_zrod, desc_cel;
    int lbajt;
9
    if (argc<3){
        fprintf(stderr, "Za malo argumentow. Uzyj:\n");
12      fprintf(stderr, "%s <plik zrodlowy> <plik docelowy>\n",         argv[0]);
        exit(1);
    }
15
    desc_zrod = open(argv[1], O_RDONLY);
```

```
71      if (desc_zrod == -1){
72  18       perror("Blad otwarcia pliku zrodlowego");
73          exit(1);
74      }
75  21
76      desc_cel = creat(argv[2], 0640);
77      if (desc_cel == -1){
78  24      perror("Blad utworzenia pliku docelowego");
79          exit(1);
80       }
81  27
82      while((lbajt = read(desc_zrod, buf, MAX)) > 0){
83          if (write(desc_cel, buf, lbajt) == -1){
84  30           perror("Blad zapisu pliku docelowego");
85              exit(1);
86          }
87  33  }
88
89      if (lbajt == -1){
90          perror("Blad odczytu pliku zrodlowego");
91  36      exit(1);
92      }
93
94  39  if (close(desc_zrod) == -1 || close(desc_cel) == -1){
95          perror("Blad zamkniecia pliku");
96          exit(1);
97  42  }
98
99      exit(0);
100 45 }
101 ################################################################################
102 Opis programu: W liniach 10-14 następuje sprawdzenie poprawności przekazania
103 argumentów z linii poleceń. Następnie otwierany jest w trybie tylko do odczytu
104 plik źródłowy i sprawdzana jest poprawność wykonania tej operacji (linie 16-20).
105 Podobnie tworzony jest i otwierany w trybie tylko do zapisu plik docelowy (linie
    22-26).
106 Właściwe kopiowanie zawartości pliku źródłowego do pliku docelowego następuje w
    pętli w linich 28-33.
107 Wyjście z pętli while następuje w wyniku zwrócenia przez funkcję read wartości 0 lub
    -1.
108 Wartość -1 oznacza błąd, co sprawdzane jest zaraz po zakończeniu pętli w liniach
    34-37.
109
110 Po każdym błędzie funkcji systemowej wyświetlany jest odpowiedni komunikat i następuje
111 zakończenie procesu przez wywołanie funkcji systemowej exit. Jeśli wywołania funkcji
112 systemowych zakończą się bezbłędnie, sterowanie dochodzi do linii 39, gdzie
    następuje zamknięcie plików.
113
114
115 Listing 2 przedstawia program do wyświetlania rozmiaru pliku. W programie
    wykorzystano funkcje
116 systemowe open, lseek i close. Nazwa pliku przykazywana jest jako argument w linii
    poleceń
117 przy uruchamianiu programu.
118
119 Listing 2: Wyprowadzanie rozmiaru pliku
120 ################################################################################
121
122     #include <fcntl.h>
123     #include <stdio.h>
124     #include <stdlib.h>
125 3
126     int main(int argc, char* argv[]){
127      int desc;
128 6    long rozm;
129
130      if (argc < 2){
131 9        fprintf(stderr, "Za malo argumentow. Uzyj:\n");
132          fprintf(stderr, "%s <nazwa pliku>\n", argv[0]);
133          exit(1);
134 12   }
135
136      desc = open(argv[1], O_RDONLY);
```

```
137   15  if (desc == -1){
138           perror("Blad otwarcia pliku");
139           exit(1);
140   18  }
141
142       rozm = lseek(desc, 0, SEEK_END);
143   21  if (rozm == -1){
144           perror("Blad w pozycjonowaniu");
145           exit(1);
146   24  }
147
148       printf("Rozmiar pliku %s: %ld\n", argv[1], rozm);
149   27
150       if (close(desc) == -1){
151           perror("Blad zamkniecia pliku");
152   30      exit(1);
153       }
154
155   33  exit(0);
156      }
157   ##############################################################################
158   Opis programu: W liniach 8-12 następuje sprawdzenie poprawności przekazania argumentów
159   z linii poleceń. Następnie otwierany jest w trybie tylko do odczytu plik o nazwie
      podanej
160   jako argument w linii poleceń i sprawdzana jest poprawność wykonania tej operacji
      (linie 14-18).
161   Po otwarciu pliku następuje przesunięcie wskaźnika bieżącej pozycji za pomocą
      funkcji lseek na
162   koniec pliku i zarazem odczyt położenia tego wskaźnika względem początku pliku
      (linia 20).
163   Uzyskany wynik działania funkcji lseek, jeżeli nie jest to wartość -1, jest
      rozmiarem pliku
164    w bajtach. Wartość ta jest wyświetlana na standardowym wyjściu (linia 26),
165    po czym plik jest zamykany (linia 28).
166   ################################################
167
168   Źródła:
169   http://wazniak.mimuw.edu.pl/
170   https://4programmers.net/C/Artykuły/Jak_programować_w_Linuksie
171
```