

Programowanie strukturalne

- wykład 4

dr Piotr Jastrzębski

Wskaźniki - cd.

System szesnastkowy

https://pl.wikipedia.org/wiki/Szesnastkowy_system_liczbowy

Pamięć wirtualna

Pamięć wirtualna – mechanizm zarządzania pamięcią komputera zapewniający procesowi wrażenie pracy w jednym, dużym, ciągłym obszarze pamięci operacyjnej podczas, gdy fizycznie może być ona pofragmentowana, nieciągła i częściowo przechowywana na urządzeniach pamięci masowej. Systemy korzystające z tej techniki ułatwiają tworzenie rozbudowanych aplikacji oraz poprawiają wykorzystanie fizycznej pamięci RAM w systemach wielozadaniowych.

ASLR

ASLR (Address Space Layout Randomization) tłumaczony jest jako mechanizm losowego generowania lokalizacji alokacji pamięci wirtualnej.

Wykonywanie czynności przedstawionych na dalszych slajdach związanych z ASLR może oznaczać narażenie komputera na niebezpieczeństwo i podatność na ataki. Wykonanie tych działań nie jest zalecane i robione tylko na własną odpowiedzialność.

Exploit Protection

Funkcja Exploit Protection jest wbudowana w system Windows 10 w celu zabezpieczenia urządzenia przed atakami. Twoje urządzenie jest od razu skonfigurowane za pomocą ustawień ochrony, które są odpowiednie dla większości użytkowników.

[Ustawienia funkcji Exploit Protection](#)

[Dowiedz się więcej](#)

Wymuś losowe generowanie obrazów (obowiązkowa funkcja ASLR)

Wymuś relokację obrazów, które nie zostały skompilowane z użyciem przełącznika /DYNAMICBASE

Włączone domyślnie



Generuj losowo alokacje pamięci (funkcja ASLR „od dołu do góry”)

Generuj losowo lokalizacje alokacji pamięci wirtualnej.

Włączone domyślnie



Funkcja ASLR o wysokiej entropii

Zwiększ zmienność podczas używania ustawienia Generuj losowo alokacje pamięci (funkcja ASLR „od dołu do góry”).

Włączone domyślnie



```
int *b;
```

równoważne zapisy:

```
int* c;
```

```
int * c;
```

```
int *c;
```

```
int*c;
```


Poprawnie:

```
int num=0;  
int *pi = &num;
```

```
int num = 0;  
int*pi;  
pi=&num;
```

Wątpliwe dla niektórych kompilatorów

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

int main()
{
    int num = 0;
    int*pi;
    pi= num;
    return 0;
}
```

```
int num = 0;
int *pi = &num;
printf("Adres pi: %d Wartosc: %d\n",&pi, pi);
printf("Adres pi: %x Wartosc: %x\n",&pi, pi);
printf("Adres pi: %o Wartosc: %o\n",&pi, pi);
printf("Adres pi: %p Wartosc: %p\n",&pi, pi);
```

Wskaźnik na stałą wartość, a stały wskaźnik

Wskaźnik na stałą wartość:

```
const int *a;  
int const * a;
```

Stały wskaźnik:

```
int * const b;
```

Stały wskaźnik na stałą wartość:

```
int const * const c
```

```
int i=0;
  const int *a=&i;
  int * const b=&i;
  int const * const c=&i;
*a = 1;  /* kompilator zaprotestuje */
*b = 2;  /* ok */
*c = 3;  /* kompilator zaprotestuje */
a = b;   /* ok */
b = a;   /* kompilator zaprotestuje */
c = a;   /* kompilator zaprotestuje */
```

Funkcja malloc

```
void *malloc(size_t size);
```

Funkcja służy do dynamicznego rezerwowania miejsca w pamięci. Gdy funkcja zostanie wywołana, w przypadku sukcesu zwróci wskaźnik do nowo zarezerwowanego miejsca w pamięci; w przypadku błędu zwraca wartość NULL.

Funkcja free

```
void free(void *ptr);
```

Funkcja free zwalnia blok pamięci wskazywany przez ptr wcześniej przydzielony przez malloc. Jeżeli ptr ma wartość NULL funkcja nie robi nic.

Rozmiar int

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

int main()
{
    printf("%Iu\n", sizeof(int)); //Windows
    //printf("%zu\n", sizeof(int)); //linux, os x
    printf("%Iu\n", sizeof(int*));
    printf("%Iu\n", sizeof(int**));
    return 0;
}
```


Wyłuskiwanie (dereferencja) wskaźnika

```
int num = 5;  
int *pi = &num;  
printf("%p\n", *pi);
```

Wskaźniki na funkcję

```
typ_zwracany (*nazwa_wsk)(typ1 arg1, typ2 arg2);
```

```
#include <stdio.h>

int suma (int lhs, int rhs)
{
    return lhs+rhs;
}

int main ()
{
    int (*wsk_suma)(int a, int b);
    wsk_suma = suma;
    printf("4+5=%d\n", wsk_suma(4,5));
    return 0;
}
```

Jaka różnica?

```
int * wsk1();  
int (*wsk2)();  
int *(*wsk3)();
```

1. Funkcja zwracająca wskaźnik.
2. Wskaźnik na funkcję.
3. Wskaźnik na funkcję, zwracającą wskaźnik.

Inne typy liczbowe?

Pełna analogia.

Duży błąd merytoryczny - 2 na egzaminie(!)

Dereferencja niezainicjalizowanych wskaźników:

```
#include <stdio.h>
int main(void)
{
    int *wsk; // niezainicjalizowany wskaźnik
    *wsk = 5;
    return 0;
}
```

Bibliografia

- ▶ Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion 2014.
- ▶ <https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki>, dostęp online 15.03.2020.
- ▶ http://wazniak.mimuw.edu.pl/index.php?title=Wst%C4%99p_do_programowania_w_j%C4%99zyku_C/Wska%C5%BAniki, dostęp online 15.03.2020.
- ▶ https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki_-_wi%C4%99cej, dostęp online 15.03.2020.