UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Sisteme de operare

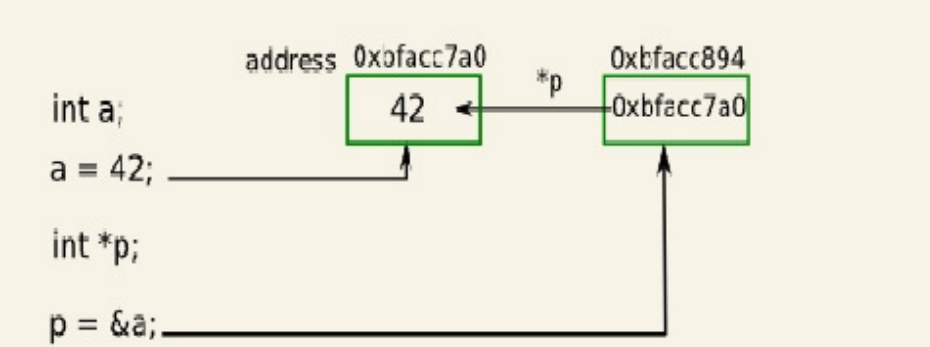
12. Zone de memorie. Gestiunea memoriei in C

Gestiunea memoriei

- Roluri ale subsitemului de gestiune a memoriei
  - ține evidența zonelor de memorie fizică (ocupate sau libere)
  - oferă proceselor sau celorlalte subsisteme acces la memorie
  - mapează paginile de memorie virtuală ale unui proces (pages) peste paginile fizice (frames)

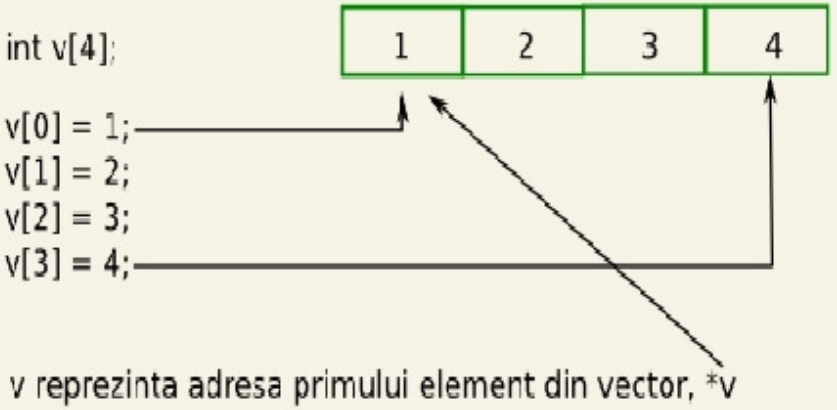
Tipuri de date

- primare (char, int) – pot reține date
- pointer – pot reține adrese din memorie
- array, struct



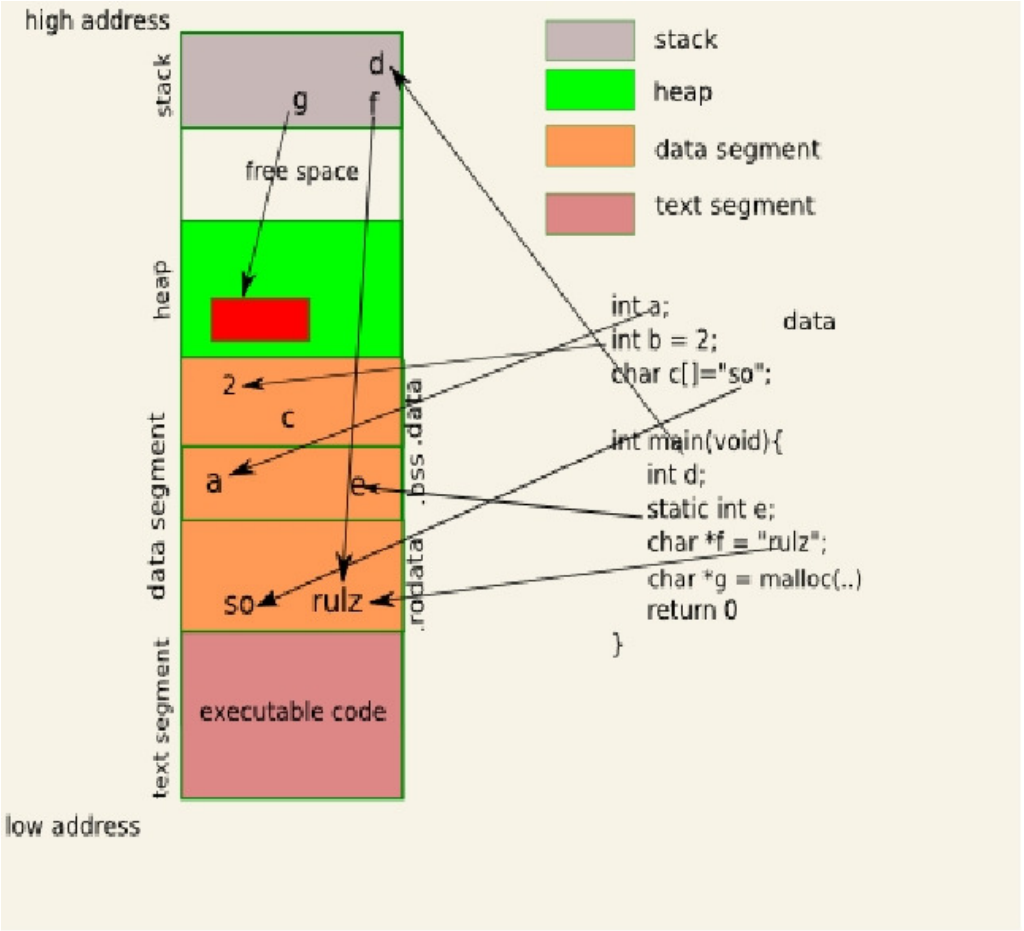
pointer - tip de date care stocheaza adresa altei variabile

int *p; - p stocheaza adresa unei variabile de tip int



Spațiul de adresă al unui proces

- Fiecare proces are propriul său spațiu de memorie virtuală
- Procesele pot partaja doar memorie fizică
- Împarțirea spațiului de adresă (1)
  - Userspace
    - utilizat în mod user
  - Kernelspace
    - Utilizat în mod kernel
    - mapat peste nucleu



Spațiul de adresă al unui proces (2)

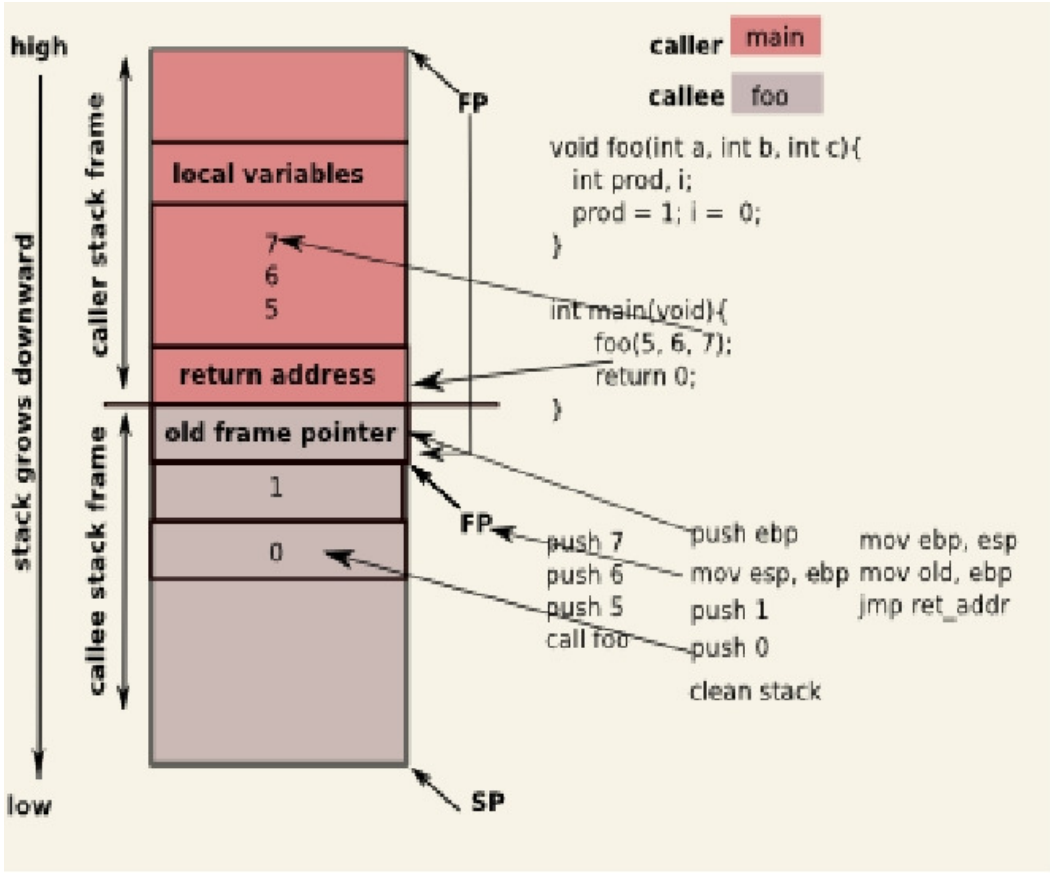
Împărțirea spațiului de adresă (2)

- zona de cod (text/segment)
  - instrucțiunile programului
  - read-only
- zone de date
  - .data – variabile statice și globale inițializate cu valori nenule
  - .bss – variabile statice și globale neinițializate sau inițializate la zero
  - .rodata – variabile read-only

Spațiul de adresă al unui proces (3)

Împărțirea spațiului de adresă (2)

- Stiva
  - regiune dinamică
  - reține stack frame-urile asociate apelurilor de funcții
  - stochează variabilele locale
  - crește în jos
  - gestionată de compilator
  - la fiecare revenire, stiva este golită



Spațiul de adresă al unui proces (3)

Împărțirea spațiului de adresă (2)

- Heap
  - regiune dinamică
  - Utilizată pentru alocarea memoriei dinamice
  - Dimensiunea regiunilor alocate este cunoscută doar la runtime
  - crește în sus
  - gestionată de programator

Alocarea memoriei

Linux

- `void *malloc(size t size);`
- `void *calloc(size t nmemb, size t size);`
- `void *realloc(void *ptr, size t size);`
- `void free(void *ptr);`

Windows

- `HANDLE HeapCreate(Options, dwInitialSize, dwMaximumSize);`
- `BOOL HeapDestroy(hHeap);`
- `LPVOID HeapAlloc(hHeap, dwFlags, dwBytes);`
- `HeapReAlloc(hHeap, dwFlags, lpMem, dwBytes);`
- `HeapFree(hHeap, dwFlags, lpMem);`

Probleme de lucru cu memoria

- memory leak-uri
  - pierderea referinței la zona de memorie

```
for(i = 0; i < 10; i++)
    a = malloc(16*sizeof(int));
free(a);
```

- accese nevalide
  - accese la zone de memorie nealocate

```
char s[4]; sprintf(s,"%s","so_rulz");
```

Probleme de lucru cu memoria (2)

- dangling reference
  - accesul la o zona de memorie care a fost anterior eliberată

```
a = malloc(16*sizeof(int));
b = a; free(b);
printf("%d", a[i]);
```

  - memoria alocată pentru `a` a fost eliberată prin intermediul lui b
- suprascrieri

Probleme de lucru cu memoria (3)

- buffer overflow
  - accese peste limita zonei de memorie alocate

```
char a[100], b[200];
scanf("%s", b);
strcpy(a, b);
```

- corupere de zone de memorie

Utilitare pentru detecția erorilor de memorie

- GDB
- mcheck
- Mtrace
- valgrind

GDB

- fișierele trebuie compilate cu opțiunea -g
- se transmite ca argument numele executabilului

  `./a.out`
- Comenzi
  - `bt`: backtrace
  - `run`:rulare
  - `step, next`: următoarea instrucțiune
  - `quit`: părăsirea depanatorului
  - `set args`: stabilirea argumentelor de rulare
  - `Disassamble`: afișează codul mașină generat de compilator
  - `info reg`: afișează conținutul registrilor
  - `man gdb`: pentru mai multe detalii

mckeck, mtrace

- mcheck
  - verifică consistența heap-ului
  - `MALLOC CHECK =1 ./executabil`
- mtrace
  - detectează memory leak-uri
  - `mtrace(), muntrace()` pe regiunea inspectată

valgrind

- suită de utilitare pentru debugging și profiling
- memcheck, callgrind, helgrind
- memcheck
  - `valgrind --tool=memcheck ./executabil`
  - detectează
    - folosirea de memorie neinițializată
    - citire/scriere din/în memorie după ce regiunea respectivă a fost eliberată
    - memory leak-uri
    - citirea/scriere dincolo de sfârșitul zonei alocate
    - folosirea necorespunzatoare a apelurilor `malloc/new` și `free/delete`
    - citirea/scrierea pe stivă în zone necorespunzătoare