

Get up to speed with this  
fun and friendly road map to the technology

# **Introduction au C**

FOR  
**DUMMIES®**

CD-ROM loaded  
with goodies

**A Reference  
for the  
Rest of Us!**

**B. Quoitin**  
**Ecole Polytechnique de Louvain**  
**Université catholique de Louvain**



# Table des matières

- Comparaison de caractères
- Tableaux dynamiques
- Structures
- Types non-standard
- Passage par adresse
- Stack et buffer overflow

# Comparaison de caractères

# Comparaison de char

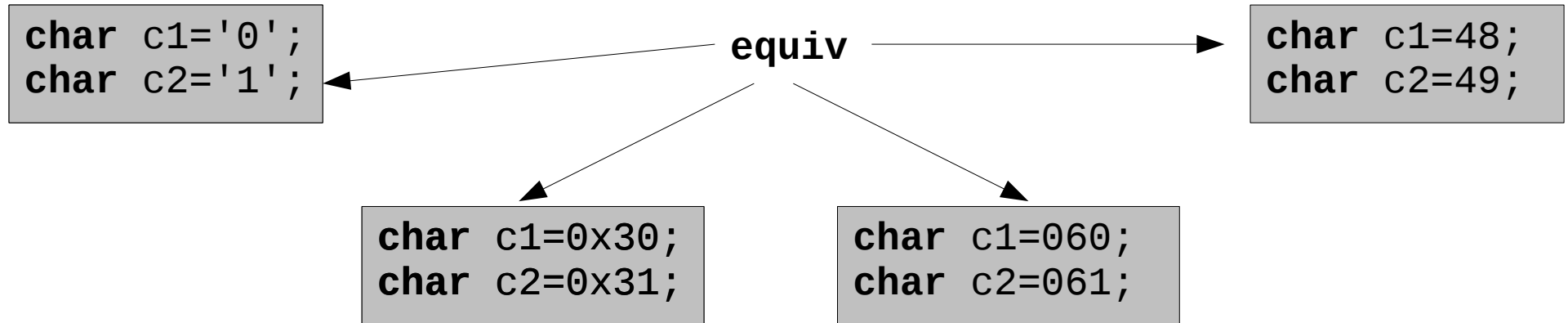
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>	86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>	87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>	88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>	89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>	90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>:</b>	91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>	92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>	93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>	94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>	95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>

Source: [www.LookupTables.com](http://www.LookupTables.com)

# Comparaison de char

```
47 2F 057 &#47; /
48 30 060 &#48; 0
49 31 061 &#49; 1
50 32 062 &#50; 2
51 33 063 &#51; 3
52 34 064 &#52; 4
53 35 065 &#53; 5
54 36 066 &#54; 6
55 37 067 &#55; 7
56 38 070 &#56; 8
57 39 071 &#57; 9
58 3A 072 &#58; :
59 3B 073 &#59; ;
60 3C 074 &#60; <
```

# Comparaison de char



`(c1 < c2)` → **TRUE**

`'9' - '0' equiv 57 - 48` → **9**

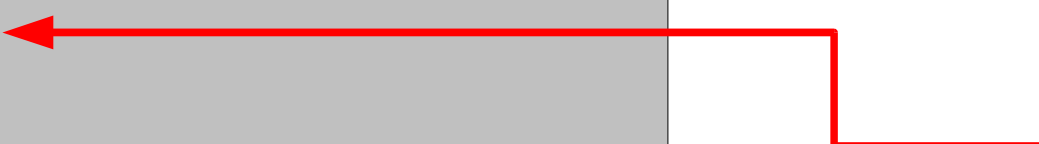
# Tableaux dynamiques

# Tableaux dynamiques

```
void nicefunct(int a)
{
    int tab[a];
    ...
}

int main(int argc, char *argv[])
{
    nicefunct(2);
}
```

## Mémoire



4072	tab
4076	
4080	adr. ret. nicefunct
4084	a: 2
4088	argv[0]: 4100
4092	argc: 1
stack: 8192	...

*Etat de la pile à l'entrée dans nicefunct*



# Tableaux dynamiques

```
void nicefunct(int a)
{
    int tab[a];
    ...
}

int main(int argc, char *argv[])
{
    nicefunct(2);
}
```

## Mémoire

4072	<del>tab</del>
4076	<del>adr. ret. nicefunct</del>
4080	a: 2
4084	argv[0]: 4100
4088	argc: 1
4092	...
stack: 8192	

*Etat de la pile à la sortie de nicefunct*

# Longueur d'une chaîne

- `strlen()` parcourt toute la chaîne jusqu'à rencontrer le `'\0'`

```
a="bonjour";  
for (i=0;i<10;i++) {  
    int l=strlen(a);  
    ...  
}
```

Pas efficace

```
a="bonjour";  
int l=strlen(a);  
for (i=0;i<10;i++) {  
    ...  
}
```

OK

- `a.length()` n'existe pas en C.

# Structures

# Structures

- Déclaration de variable composée

```
struct  
{  
    ( type identifiant; )+  
} identifiant ;
```

- Déclaration de structure

```
struct identifiant_struct  
{  
    ( type identifiant; )+  
};
```

# Structures

- Exemple (variable composée)

```
#define MAX_LONG_NOM 40
#define MAX_LONG_REM 100
struct
{
    char        nom[MAX_LONG_NOM];
    char        remarque[MAX_LONG_DESCR];
    unsigned int resultat;
} etudiant;

strncpy(etudiant.nom, "C. Pecheur", MAX_LONG_NOM);
strncpy(etudiant.remarque, "Confond fréquemment Java et C",
        MAX_LONG_REM);
etudiant.resultat= 12;
```

# Structures

- Exemple

```
#define MAX_LONG_NOM 40
#define MAX_LONG_REM 100
struct type_etudiant
{
    char          nom[MAX_LONG_NOM];
    char          remarque[MAX_LONG_DESCR];
    unsigned int  resultat;
};

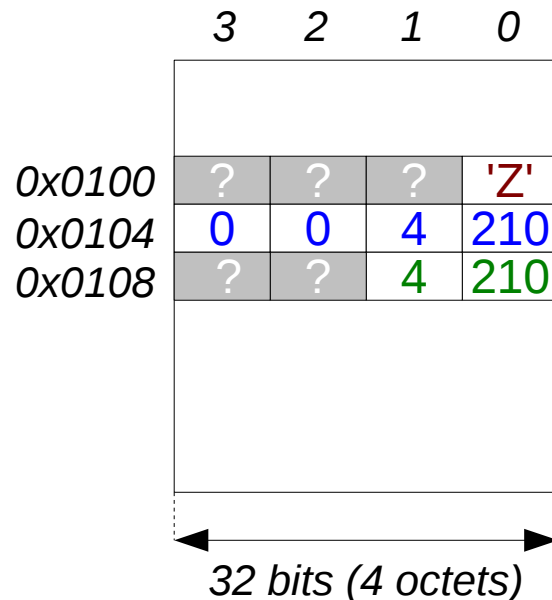
struct type_etudiant etudiant1;
struct type_etudiant etudiant2;
strncpy(etudiant1.nom, "C. Pecheur", MAX_LONG_NOM);
strncpy(etudiant1.remarque, "Confond fréquemment Java et C",
        MAX_LONG_REM);
etudiant1.resultat= 12;

strncpy(etudiant2.nom, "Y. Deville", MAX_LONG_NOM);
/* ... */
```

# Structures

- Alignement en mémoire

```
struct
{
    char    champ_1;
    int     champ_2;
    short int champ_3;
} alignee = { 'Z', 1234, 1234 };
```



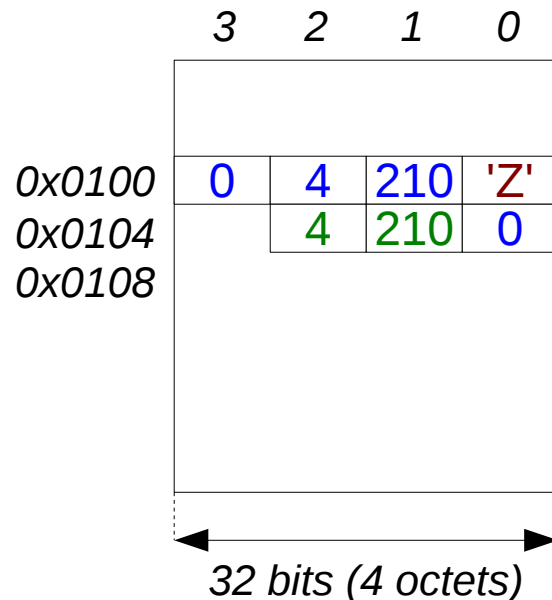
**sizeof(alignee)**  
vaut 12

? = « padding »

# Structures

- Alignement en mémoire

```
struct __attribute__((packed))  
{  
    char    champ_1;  
    int     champ_2;  
    short int champ_3;  
} non_alignee = { 'Z', 1234, 1234 };
```



**sizeof(non\_alignee)**  
vaut 7



# Types non-standard

# Types non-standard

- Déclaration

```
typedef type identifiant_type ;
```

- Exemples

```
typedef char * pointeur_vers_char;  
pointeur_vers_char str= "Plop-plop";  
  
typedef struct {  
    int x, y;  
} type_cooronnees;  
  
type_cooronnees coord= { 50, 4 };
```

# Passage par adresse

# Passage par adresse

- Rappel: passage par valeur

```
void fct(int a)
{
    a= 10;
}

int main()
{
    int b= 5;
    fct(b);
    print("Valeur de b:%d\n", b);
    return 0;
}
```

# Passage par adresse

- Passage par adresse

```
void fct(int * a)
{
    *a= 10;
}

int main()
{
    int b= 5;
    fct(&b);
    print("Valeur de b:%d\n", b);
    return 0;
}
```

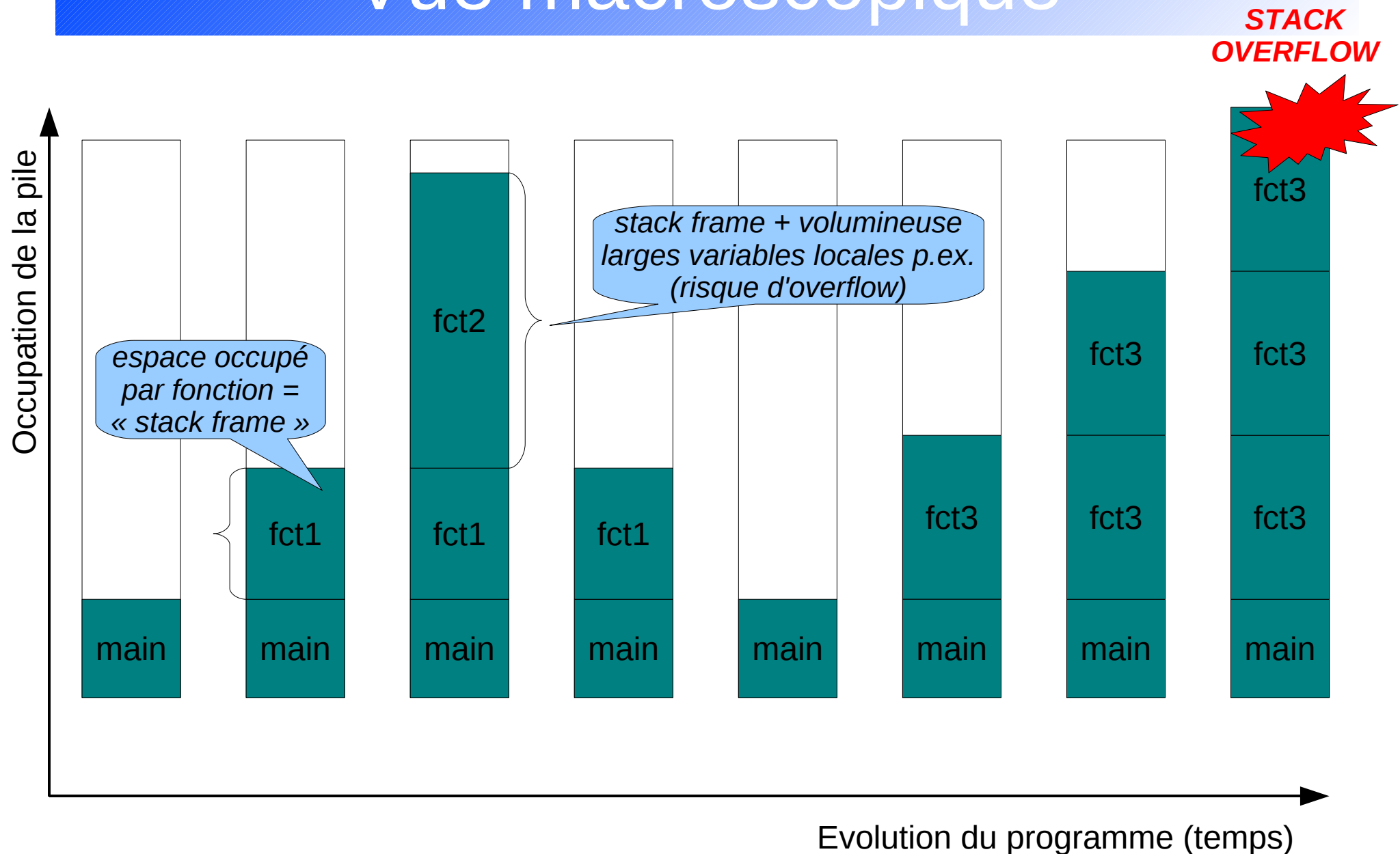
# Passage par adresse

- Passage de pointeur par adresse

```
void fct(char * str, char ** zeroptr)
{
    *zeroptr= NULL;
    while (*str != '\0') {
        if (*str == '0') {
            *zeroptr= str;
            break;
        }
        str++;
    }
}
```

```
int main()
{
    char * str= "123045";
    char * zero;
    fct(str, &zero);
    if (zero != NULL)
        print("Position du 1er zéro:%d\n",
            zero-str);
    else
        print("Pas de zéro trouvé\n");
    return 0;
}
```

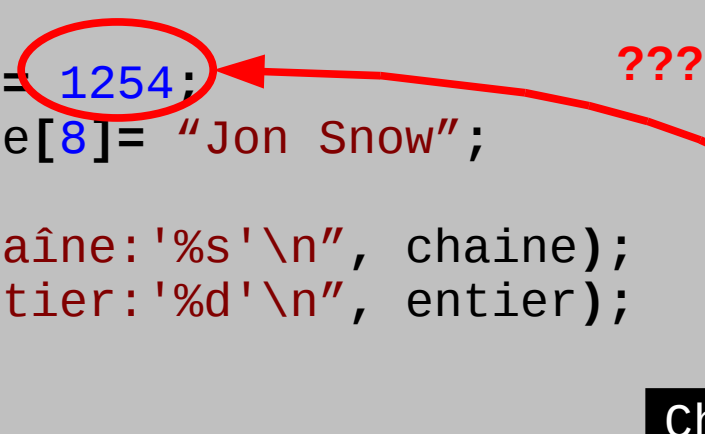
# Vue macroscopique



# Buffer overflow

```
int main()
{
    int entier= 1254;
    char chaine[8]= "Jon Snow";

    printf("Chaîne: '%s'\n", chaine);
    printf("Entier: '%d'\n", entier);
    return 0;
}
```



Chaîne: 'Jon Snow'  
Entier: 1024

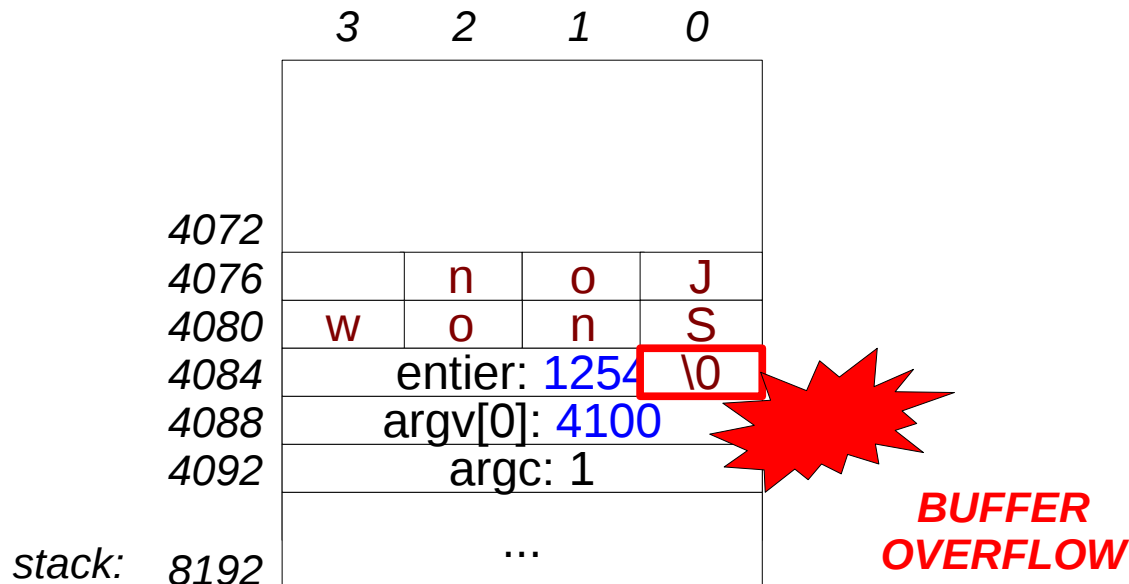


# Buffer overflow

```
int main()
{
    int entier= 1254;
    char chaine[8]= "Jon Snow";

    printf("Chaîne: '%s'\n", chaine);
    printf("Entier: '%d'\n", entier);
    return 0;
}
```

Chaîne: 'Jon Snow'  
Entier: 1024



# FIN

## Questions ?