Sistemas Operativos / Operating Systems

Aula P05
The C Language

Carlos Coutinho

Carlos.Eduardo.Coutinho@iscte-iul.pt

Topics

- Read strings: fgets() vs scanf()
- Vector addresses

Read strings: basic scanf()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf(“%s %s”, str1, str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Read strings: basic scanf()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf(“%s %s”, str1, str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

```
Write two strings:
123
456
Result: [123][456]
```

Read strings: basic scanf()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf(“%s %s”, str1, str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

```
Write two strings:
123
456
Result: [123][456]
```

```
Write two strings:
123 456
Result: [123][456]
```

Read strings: basic scanf()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf(“%s %s”, str1, str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

```
Write two strings:
123
456
Result: [123][456]
```

```
Write two strings:
123 456
Result: [123][456]
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
"123 456" 789
Result: ["123][456"]
```

✗

Read strings: basic scanf()

```c
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf("Write two strings:\n");
    scanf("%s %s", str1, str2);
    printf("Result: [%s][%s]\n", str1, str2);
}
```

```
Write two strings:
123
456
Result: [123][456]
```

```
Write two strings:
123 456
Result: [123][456]
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
"123 456" 789
Result: ["123][456"]
```

✗

Problem 2: **scanf()** allows buffer overrun:

```
Write two strings:
123
textomuitolongo
Result: [muitolongo][textomuitolongo]
```

✗

Read strings: Solution candidate: gets()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    gets(str1);
    gets(str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

Read strings: Solution candidate: gets()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    gets(str1);
    gets(str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

Read strings: Solution candidate: gets()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    gets(str1);
    gets(str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

gets() solves the problem ✓

Read strings: Solution candidate: gets()

```c
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    gets(str1);
    gets(str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

gets() solves the problem ✓

Problem 2: **scanf()** allows buffer overrun:

Read strings: Solution candidate: gets()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    gets(str1);
    gets(str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

gets() solves the problem ✓

Problem 2: **scanf()** allows buffer overrun:

Problem: **gets()** also allows buffer overrun:

```
Write two strings:
123
textomuitolongo
Result: [muitolongo][textomuitolongo]
```

✗

Read strings: Solution candidate: fgets()

Problem 1: **scanf()** always stops on every separator, even space:

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    fgets(str1, 10, stdin);
    fgets(str2, 5, stdin);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Read strings: Solution candidate: fgets()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    fgets(str1, 10, stdin);
    fgets(str2, 5, stdin);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456
][txt
]
```

fgets() solves the problem ✓

Read strings: Solution candidate: fgets()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    fgets(str1, 10, stdin);
    fgets(str2, 5, stdin);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456
][txt
]
```

fgets() solves the problem ✓

Problem 2: **scanf()** allows buffer overrun:

Problem: **gets()** also allows buffer overrun:

Read strings: Solution candidate: fgets()

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    fgets(str1, 10, stdin);
    fgets(str2, 5, stdin);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456
][txt
]
```

fgets() solves the problem ✓

Problem 2: **scanf()** allows buffer overrun:

Problem: **gets()** also allows buffer overrun:

```
Write two strings:
123
textomuitolongo
Result: [123
][text]
```

fgets() solves the problem ✓

Read strings: Solution candidate: fgets()

```c
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    fgets(str1, 10, stdin);
    fgets(str2, 5, stdin);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

```
Write two strings:
123 456
txt
Result: [123 456
][txt
]
```

str1

	‘1’	‘2’	‘3’	‘ ’	‘4’	‘5’	‘6’	‘\n’	‘\0’		

str2

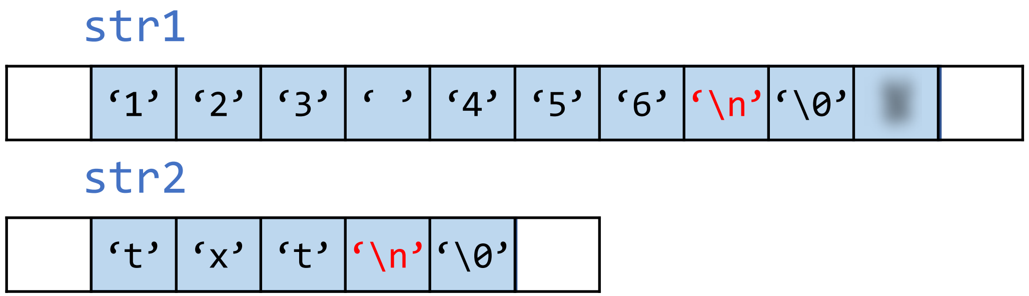
	‘t’	‘x’	‘t’	‘\n’	‘\0’	

Problem: **fgets()** reads all until \n… **INCLUDING the \n**:

Read strings: Solution 1: fgets() with workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    fgets(str1, 10, stdin);
    fgets(str2, 5, stdin);
    printf(“Result1: [%s][%s]\n”, str1, str2);
    str1[strlen(str1) – 1] = ‘\0’;
    str2[strlen(str2) – 1] = ‘\0’;
    printf(“Result2: [%s][%s]\n”, str1, str2);
}
```

```
Write two strings:
123 456
txt
Result1: [123 456
][txt
]
```



str1

‘1’ ‘2’ ‘3’ ‘ ’ ‘4’ ‘5’ ‘6’ ‘\n’ ‘\0’

str2

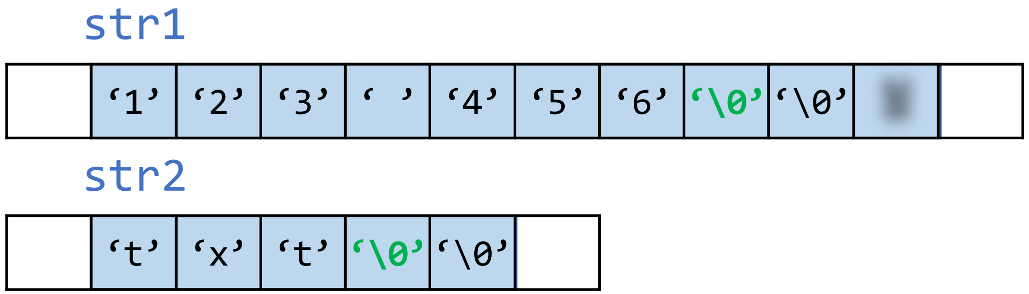
‘t’ ‘x’ ‘t’ ‘\n’ ‘\0’

Problem: **fgets()** reads all until \n… **INCLUDING the \n**:

Read strings: Solution 1: fgets() with workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    fgets(str1, 10, stdin);
    fgets(str2, 5, stdin);
    printf(“Result1: [%s][%s]\n”, str1, str2);
    str1[strlen(str1) – 1] = ‘\0’;
    str2[strlen(str2) – 1] = ‘\0’;
    printf(“Result2: [%s][%s]\n”, str1, str2);
}
```

```
Write two strings:
123 456
txt
Result1: [123 456
][txt
]
```



str1

	‘1’	‘2’	‘3’	‘ ’	‘4’	‘5’	‘6’	‘\0’	‘\0’		

str2

	‘t’	‘x’	‘t’	‘\0’	‘\0’	

Problem: **fgets()** reads all until \n… **INCLUDING the \n**:

```
Write two strings:
123 456
txt2
Result1: [123 456
][txt
]
Result2: [123 456][txt]
```

✓

fgets() with workaround solves the problem

Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

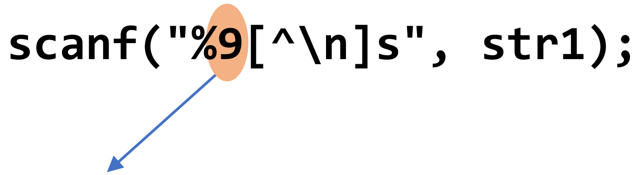
Problem 1: **scanf()** always stops on every separator, even space:

```
scanf("%9[^\n]s", str1);
```

Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf("Write two strings:\n");
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf("Result: [%s][%s]\n", str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:



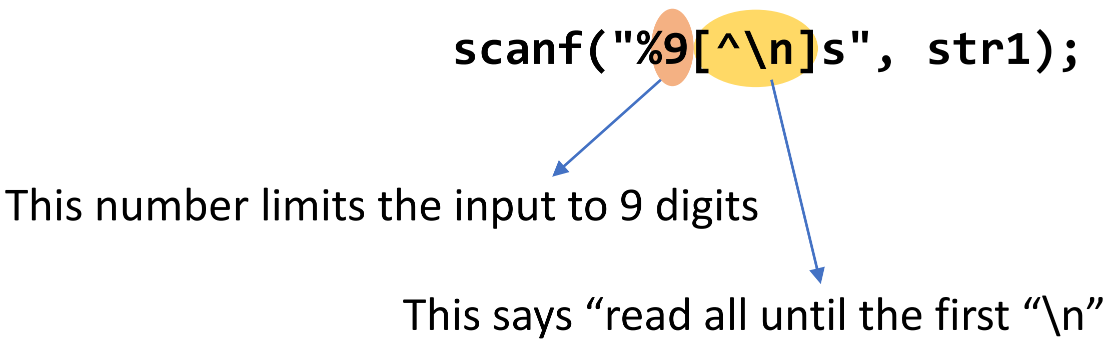
```
scanf("%9[^\n]s", str1);
```

This number limits the input to 9 digits

Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:



```
scanf("%9[^\n]s", str1);
```

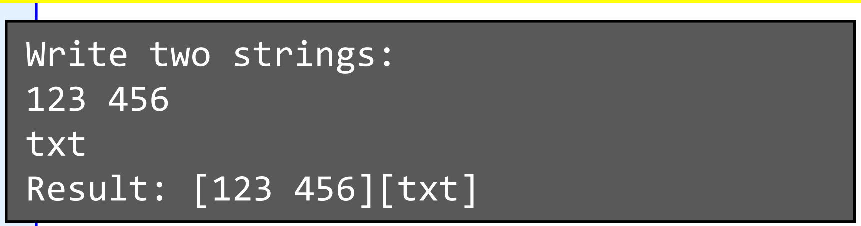
This number limits the input to 9 digits

This says “read all until the first “\n”

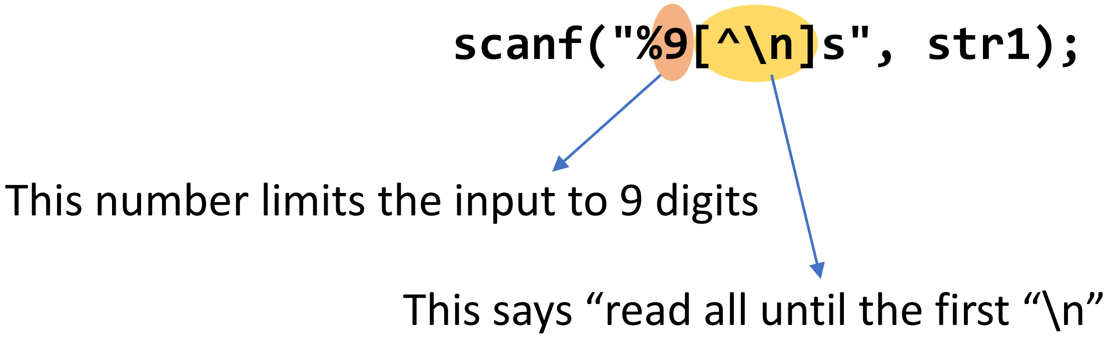
Read strings: Solution candidate: scanf() w/ workaround

```c
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:



```
Write two strings:
123 456
txt
Result: [123 456][txt]
```



```c
scanf("%9[^\n]s", str1);
```

This number limits the input to 9 digits

This says “read all until the first “\n”

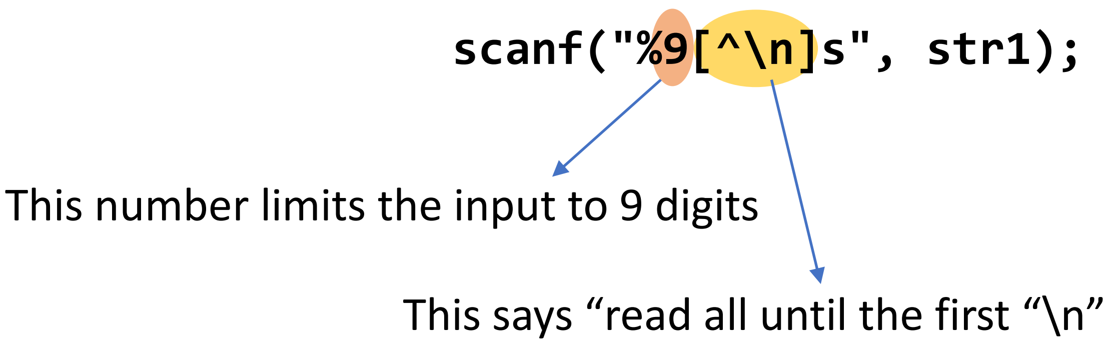
Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

scanf() with workaround solves the problem ✓



`scanf("%9[^\n]s", str1);`

This number limits the input to 9 digits

This says “read all until the first “\n”

Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

scanf() with workaround solves the problem ✓

Problem 2: **scanf()** allows buffer overrun:

Problem: **gets()** also allows buffer overrun:

Read strings: Solution candidate: scanf() w/ workaround

```c
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

scanf() with workaround solves the problem ✓

Problem 2: **scanf()** allows buffer overrun:

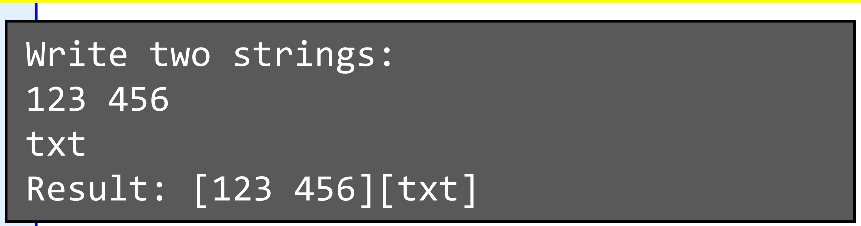
Problem: **gets()** also allows buffer overrun:

```
Write two strings:
123
textomuitolongo
Result: [123][text]
```

Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

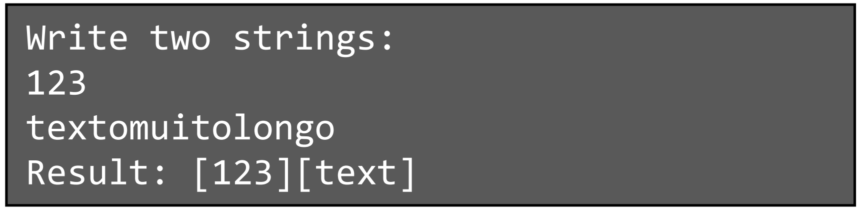


```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

scanf() with workaround solves the problem ✓

Problem 2: **scanf()** allows buffer overrun:

Problem: **gets()** also allows buffer overrun:



```
Write two strings:
123
textomuitolongo
Result: [123][text]
```

scanf() with workaround solves the problem ✓

Read strings: Solution candidate: scanf() w/ workaround

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

scanf() with workaround solves the problem ✓

Problem: if the **first** string is too big, **scanf()** doesn’t ask for the second

```
Write two strings:
1234567890123
Result: [123456789][0123]
```

Read strings: Solution 2: scanf() w/ workaround & clear

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%*[^\n]"); // Clean stdin buffer
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

scanf() with workaround solves the problem ✓

Problem: if the **first** string is too big, **scanf()** doesn’t ask for the second

```
Write two strings:
1234567890123
txt
Result: [123456789][txt]
```

Read strings: Solution 2: scanf() w/ workaround & clear

```
#include <stdio.h>

int main() {
    char str1[10], str2[5];
    printf(“Write two strings:\n”);
    scanf("%9[^\n]s", str1);
    scanf("%*[^\n]"); // Clean stdin buffer
    scanf("%4[^\n]s", str2);
    printf(“Result: [%s][%s]\n”, str1, str2);
}
```

Problem 1: **scanf()** always stops on every separator, even space:

```
Write two strings:
123 456
txt
Result: [123 456][txt]
```

scanf() with workaround solves the problem ✓

Problem: if the **first** string is too big, **scanf()** doesn’t ask for the second

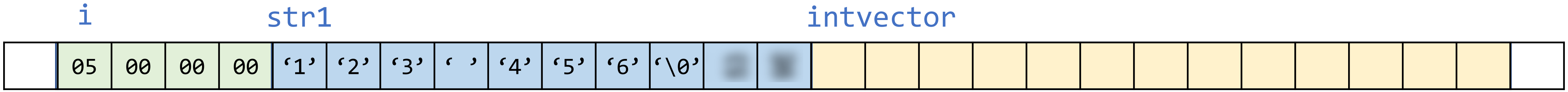
```
Write two strings:
1234567890123
txt
Result: [123456789][txt]
```

scanf() with workaround and clear stdin buffer solves the problem ✓

Vector addresses

```
#include <stdio.h>

int function1(int *pi) {
    ...
}

int main() {
    int i = 5;
    char str1[10] = “123 456”;
    int intvector[5];
    scanf("%d", &i);          // Requires &
    scanf("%9[^\n]s", str1);  // Does not use &
    i = function1(intvector); // Does not use &
}
```

Why?



Vector addresses

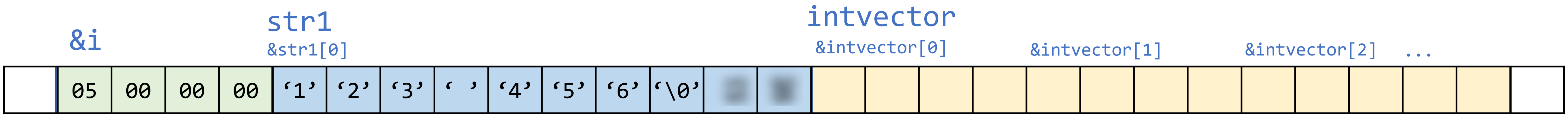
```
#include <stdio.h>

int function1(int *pi) {
    ...
}

int main() {
    int i = 5;
    char str1[10] = “123 456”;
    int intvector[5];
    scanf("%d", &i);            // Requires &
    scanf("%9[^\n]s", str1);  // Does not use &
    i = function1(intvector); // Does not use &
}
```

Why?

because actually,

`str1 == &str1[0]`

`intvector == &intvector[0]`


&i
str1
&str1[0]
intvector
&intvector[0]
&intvector[1]
&intvector[2]
...
05
00
00
00
‘1’
‘2’
‘3’
‘ ’
‘4’
‘5’
‘6’
‘\0’