Access Control and Privacy Policies (10)

Email: christian.urban at kcl.ac.uk
Office: S1.27 (1st floor Strand Building)

Slides: KEATS (also homework is there)

Revision

1st Lecture

- hashes and salts to guaranty data integrity
- storing passwords (brute force attacks and dictionary attacks)

2nd Lecture: E-Voting

- Integrity
- Ballot Secrecy
- Voter Authentication
- Enfranchisement
- Availability

2nd Lecture: E-Voting

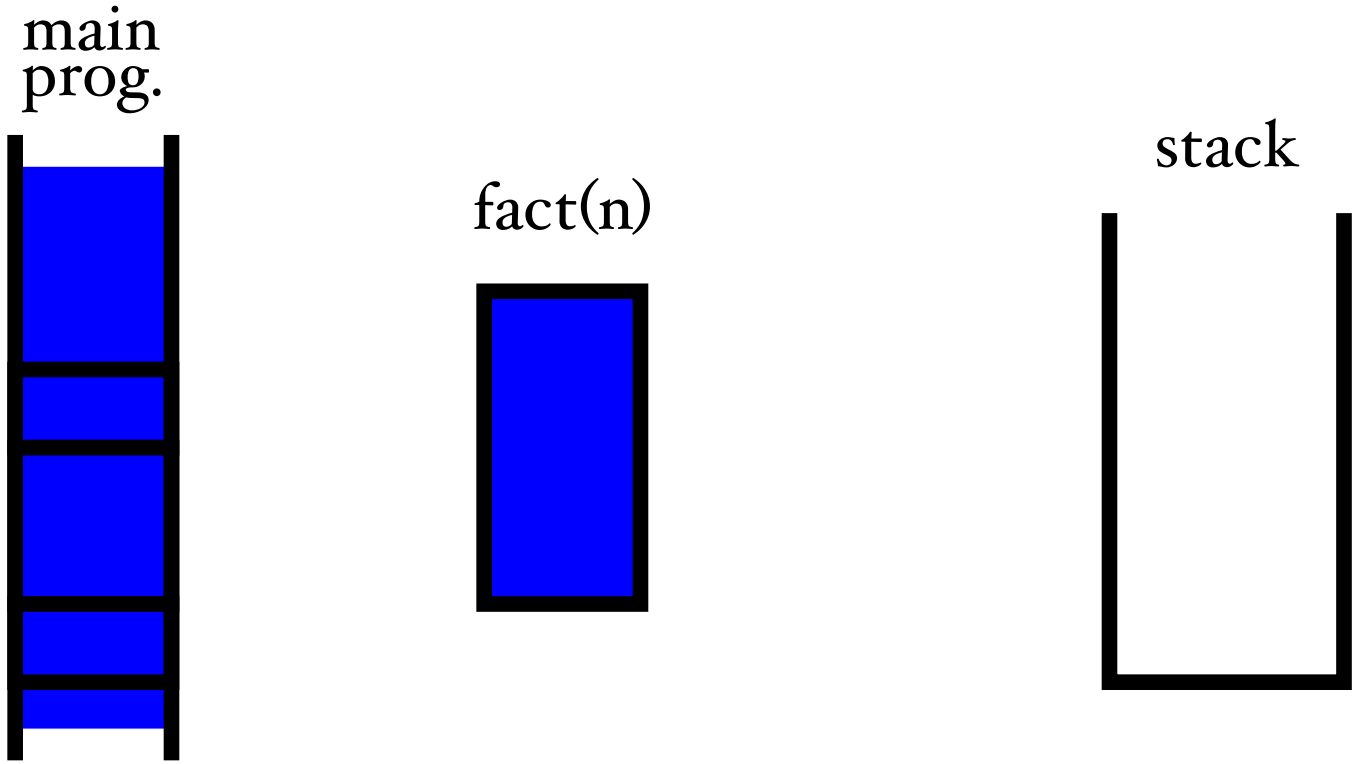
Online Banking vs. E-Voting

- online banking: if fraud occurred you try to identify who did what (somebody's account got zero)

- e-voting: some parts can be done electronically, but not the actual voting (final year project: online voting)

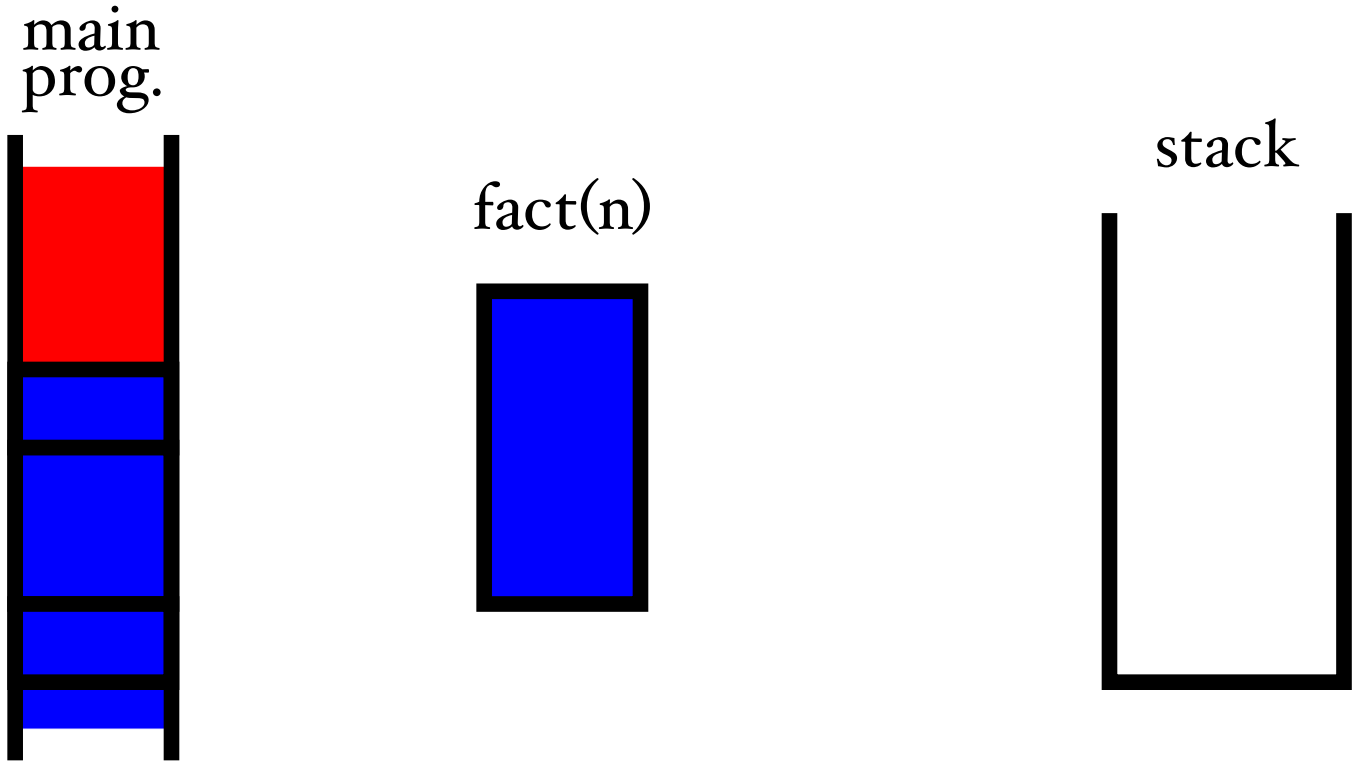
3rd Lecture: Buffer Overflow Attacks

- the problem arises from the way C/C++ organises its function calls



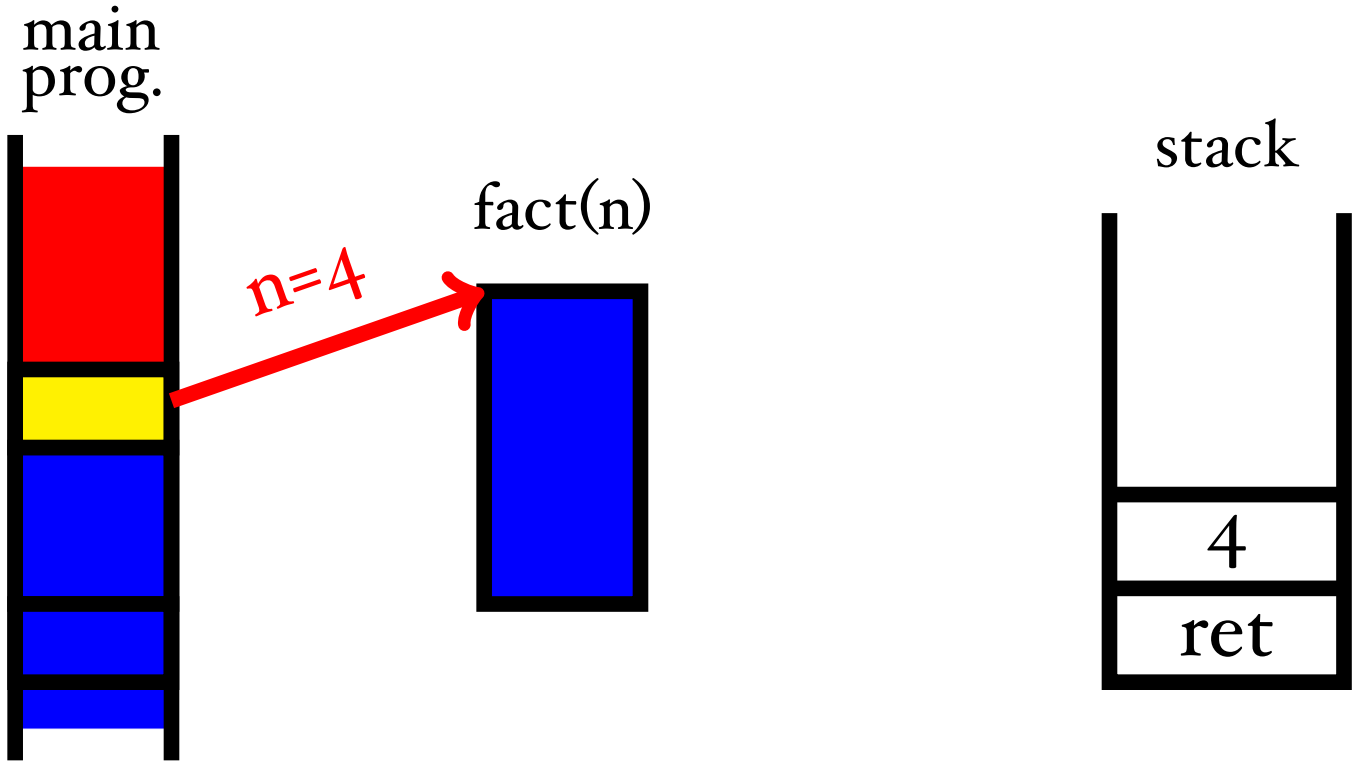
3rd Lecture: Buffer Overflow Attacks

- the problem arises from the way C/C++ organises its function calls



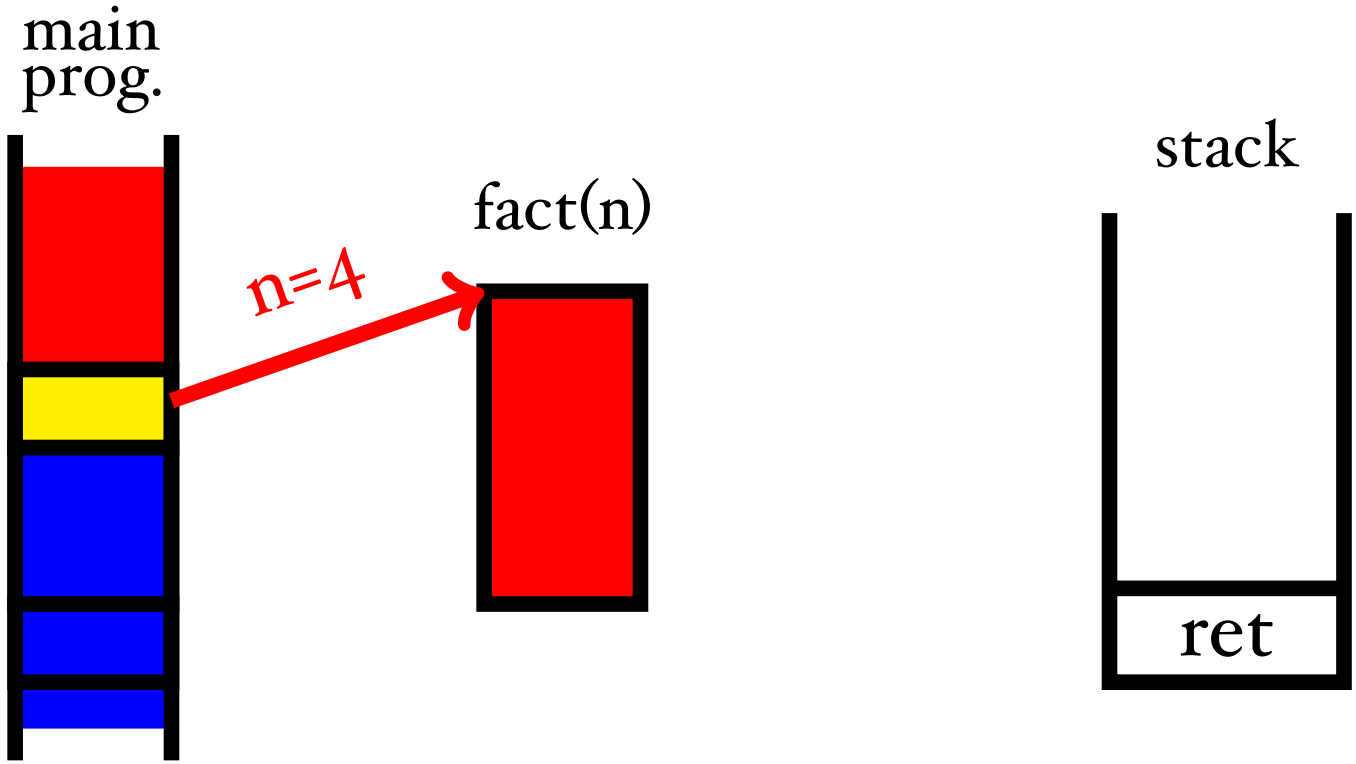
3rd Lecture: Buffer Overflow Attacks

- the problem arises from the way C/C++ organises its function calls



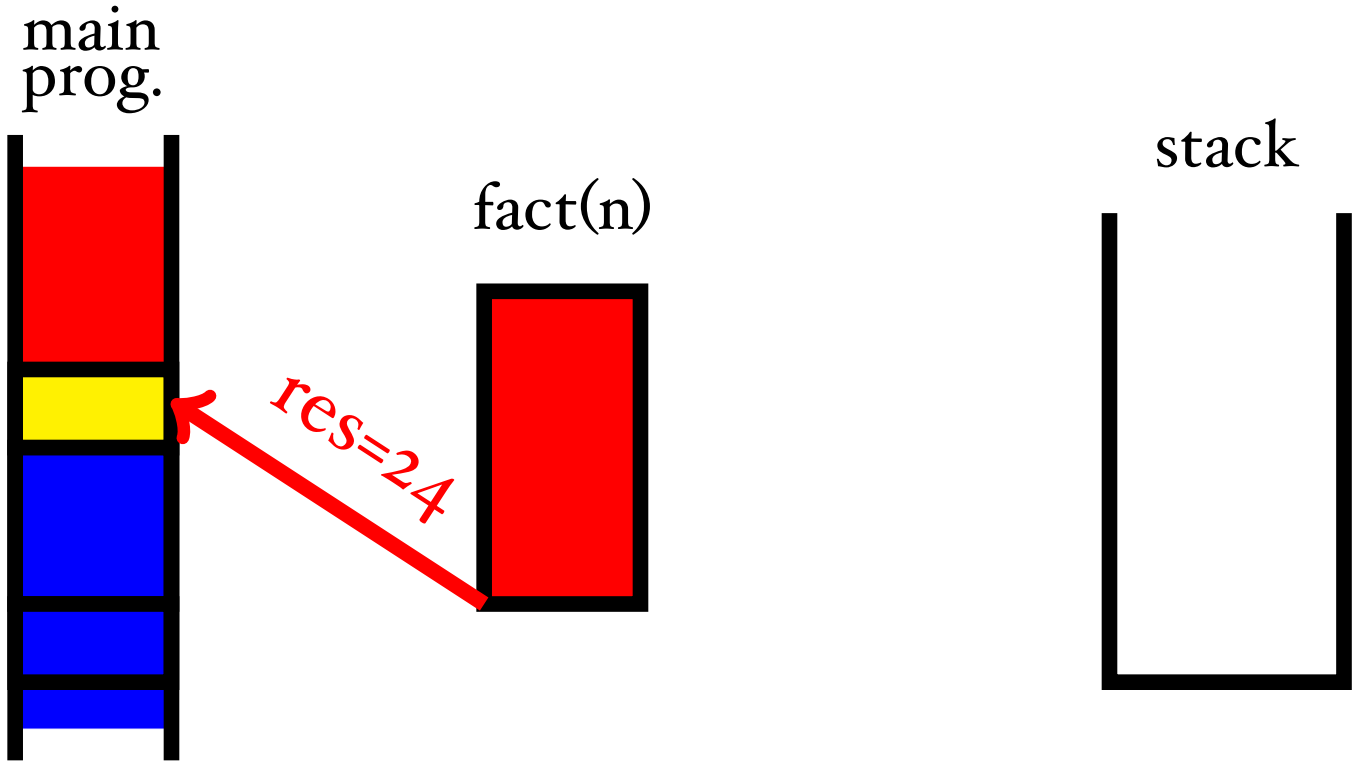
3rd Lecture: Buffer Overflow Attacks

- the problem arises from the way C/C++ organises its function calls



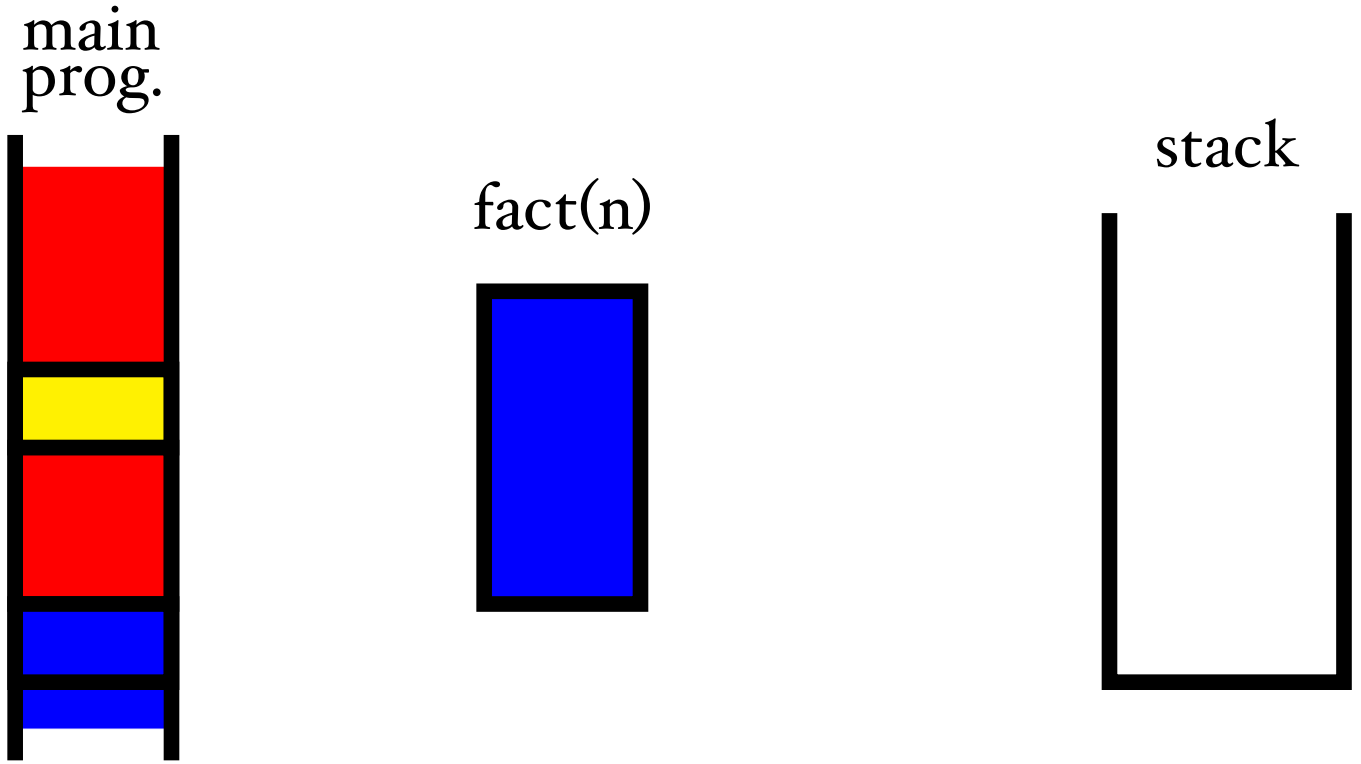
3rd Lecture: Buffer Overflow Attacks

- the problem arises from the way C/C++ organises its function calls



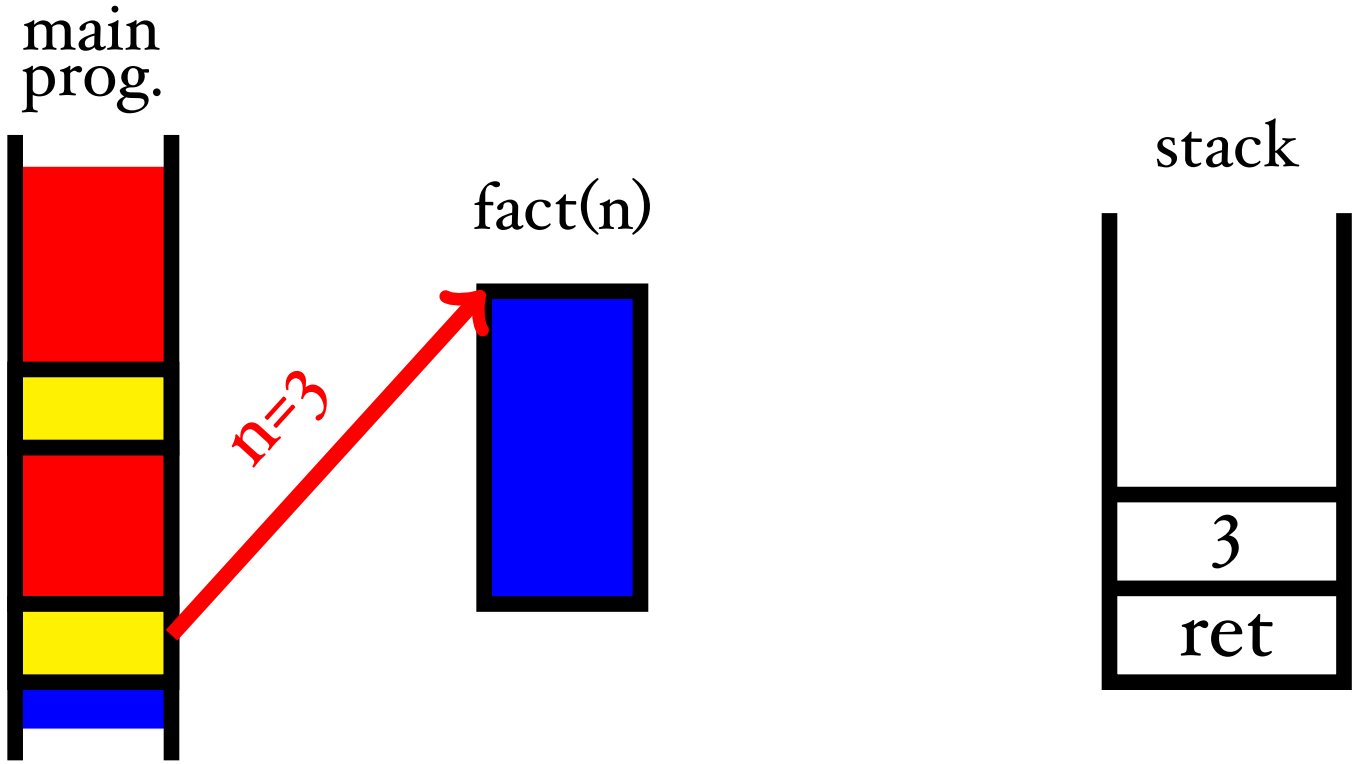
3rd Lecture: Buffer Overflow Attacks

- the problem arises from the way C/C++ organises its function calls



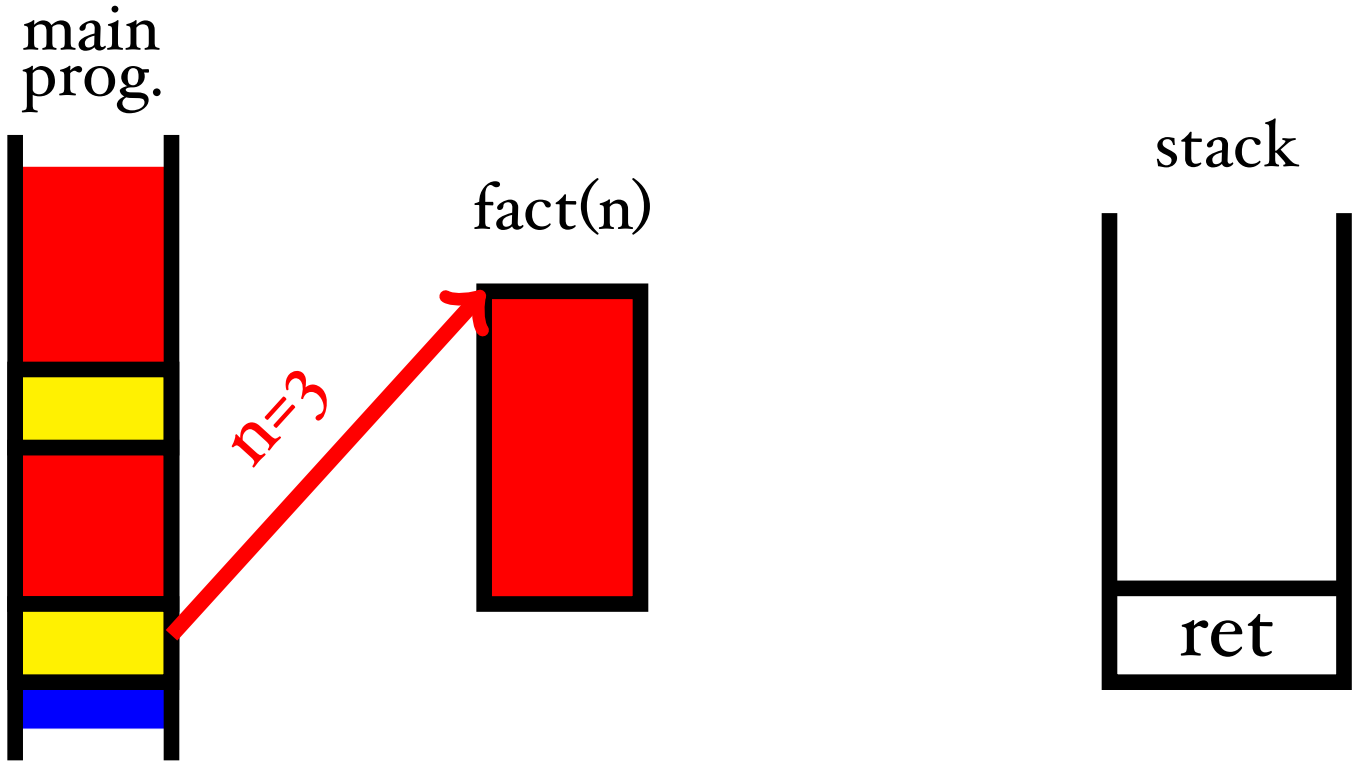
3rd Lecture: Buffer Overflow Attacks

- the problem arises from the way C/C++ organises its function calls



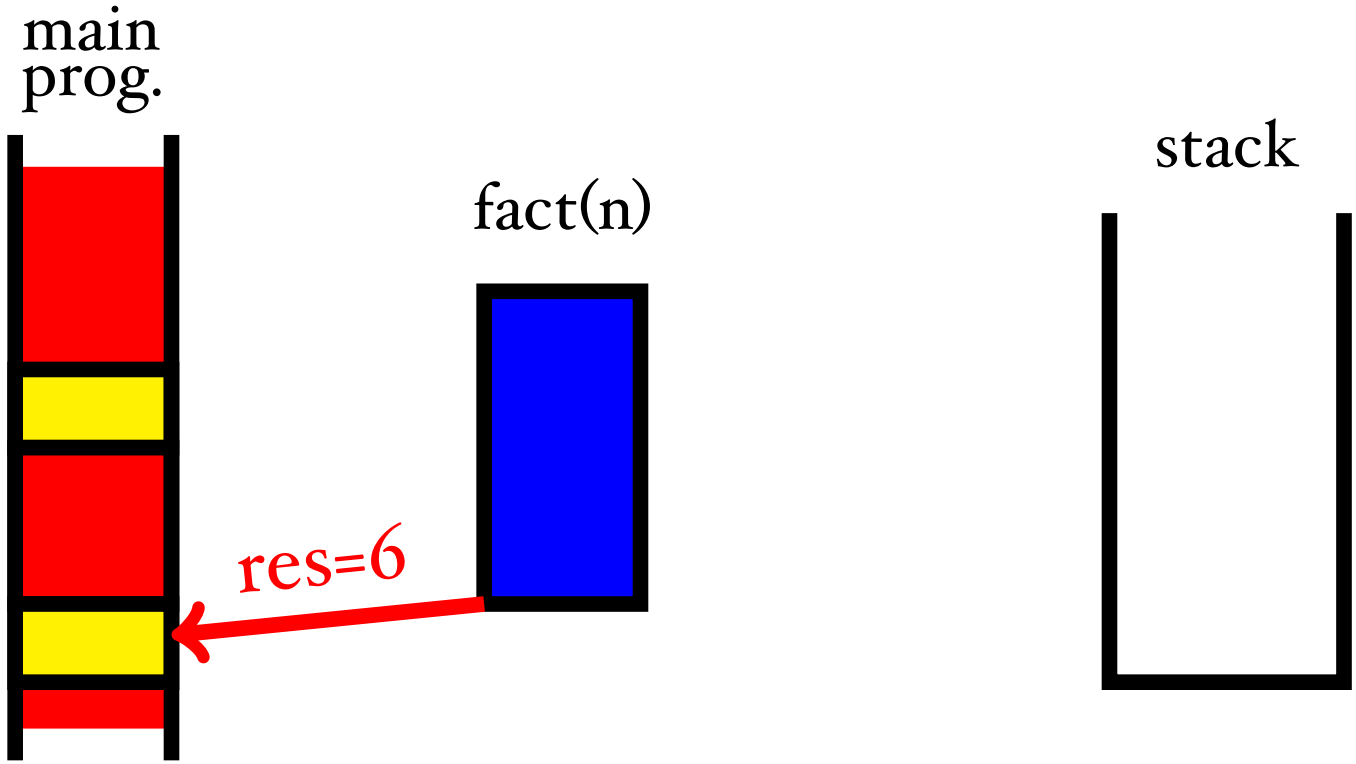
3rd Lecture: Buffer Overflow Attacks

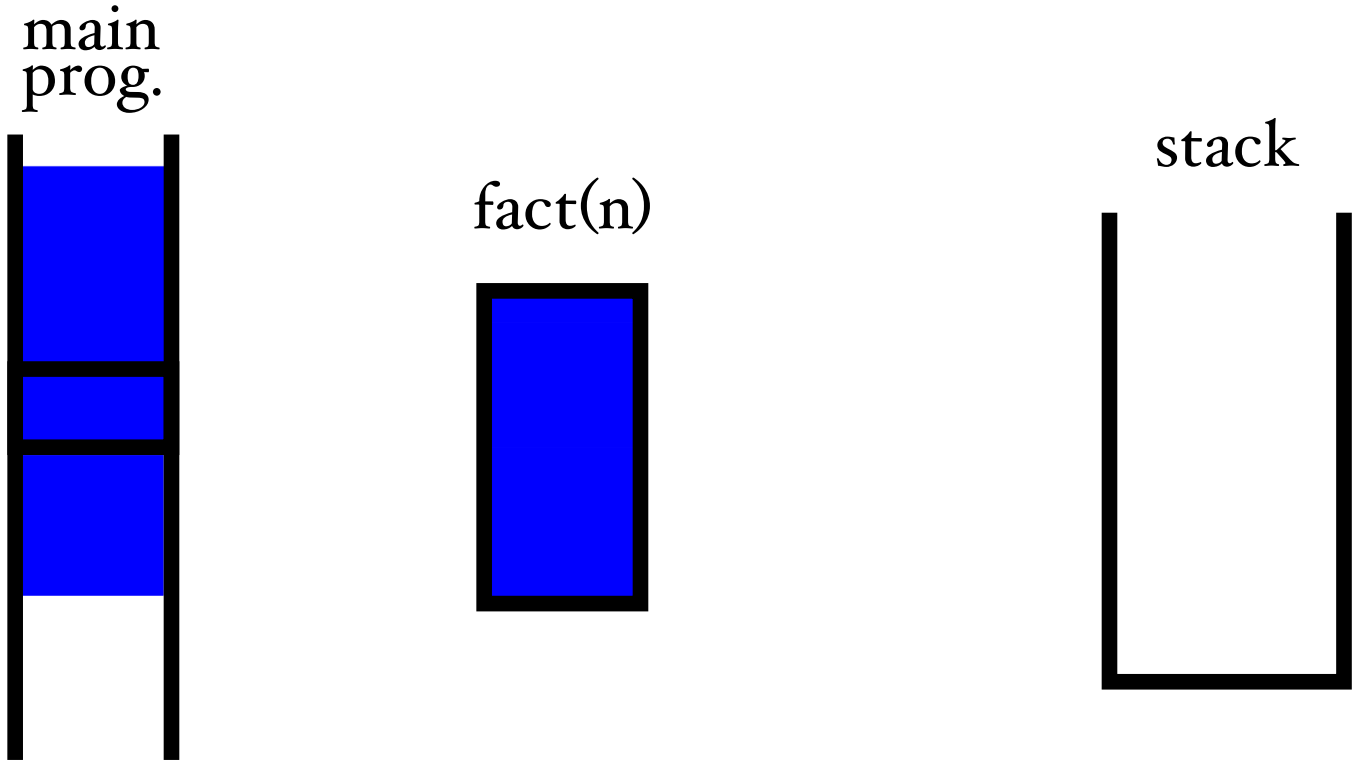
- the problem arises from the way C/C++ organises its function calls

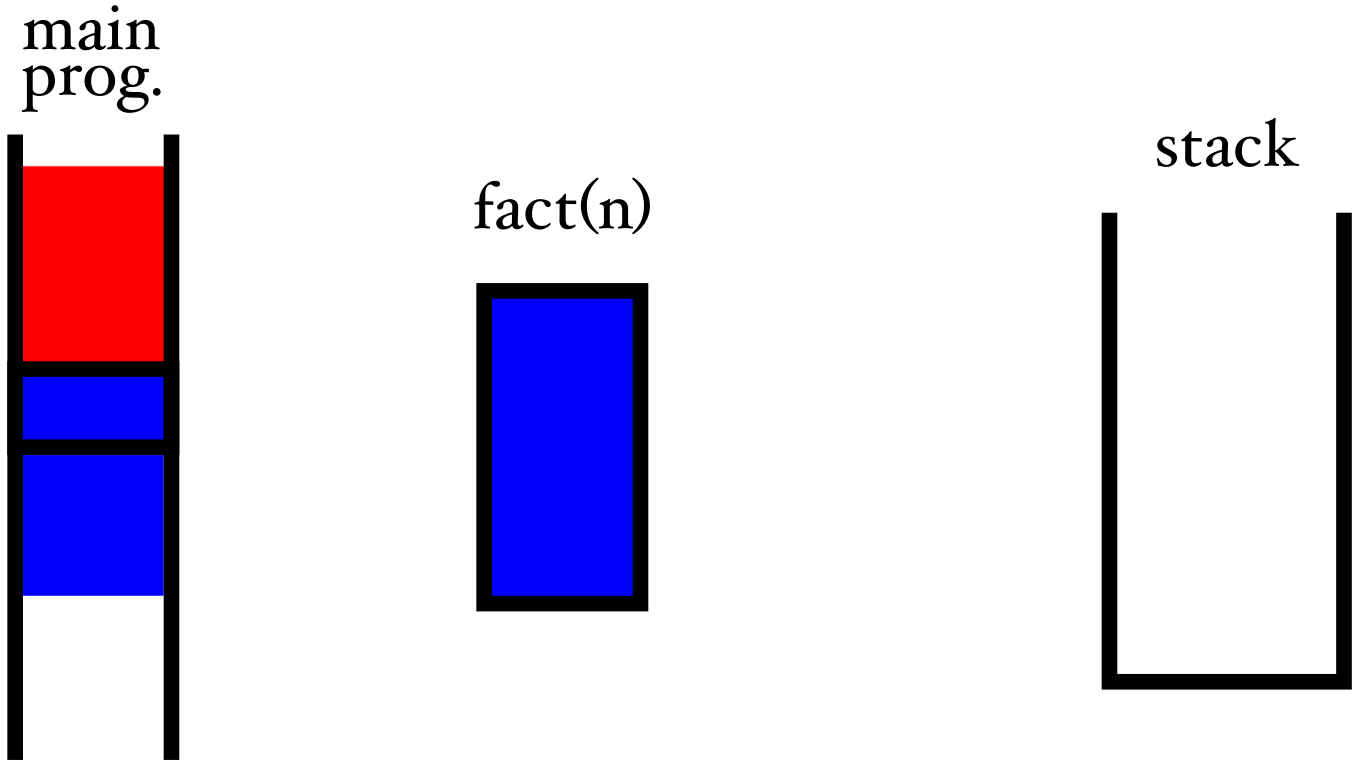


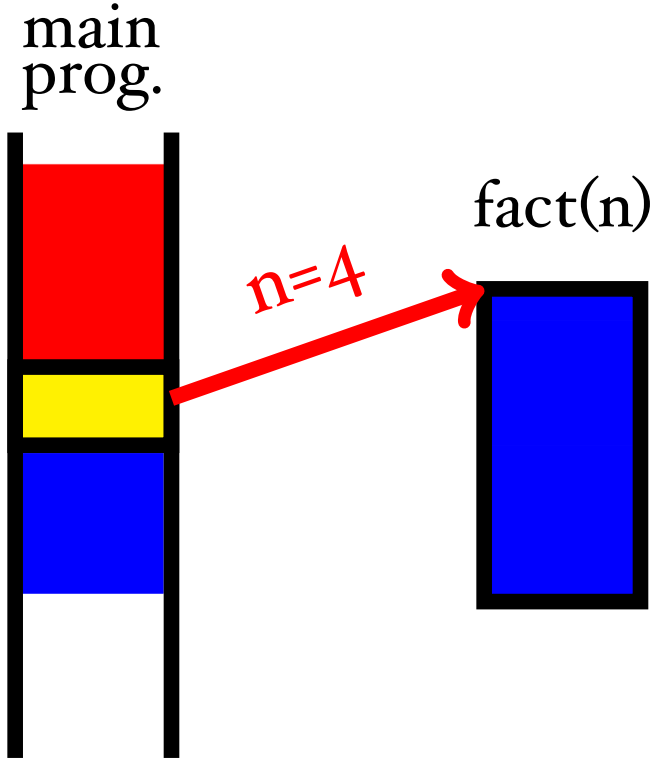
3rd Lecture: Buffer Overflow Attacks

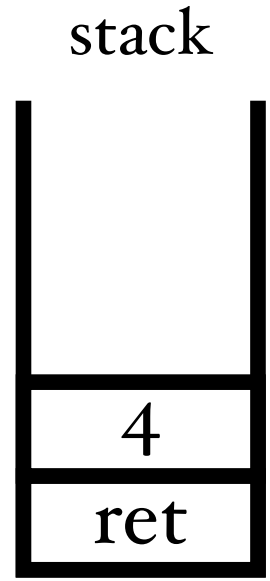
- the problem arises from the way C/C++ organises its function calls

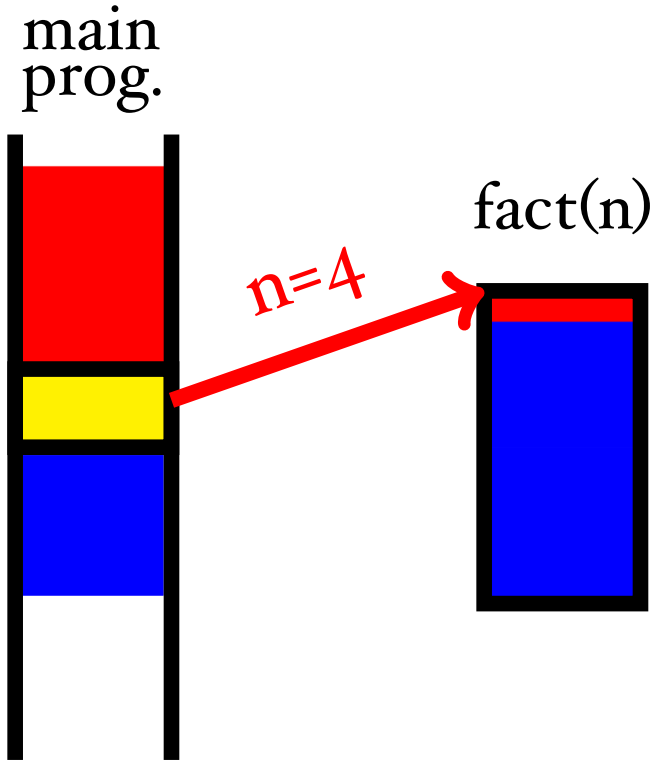


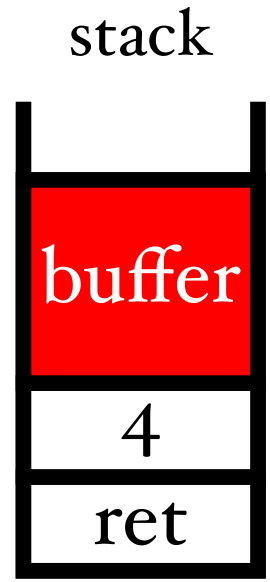

main
prog.
fact(n)
stack

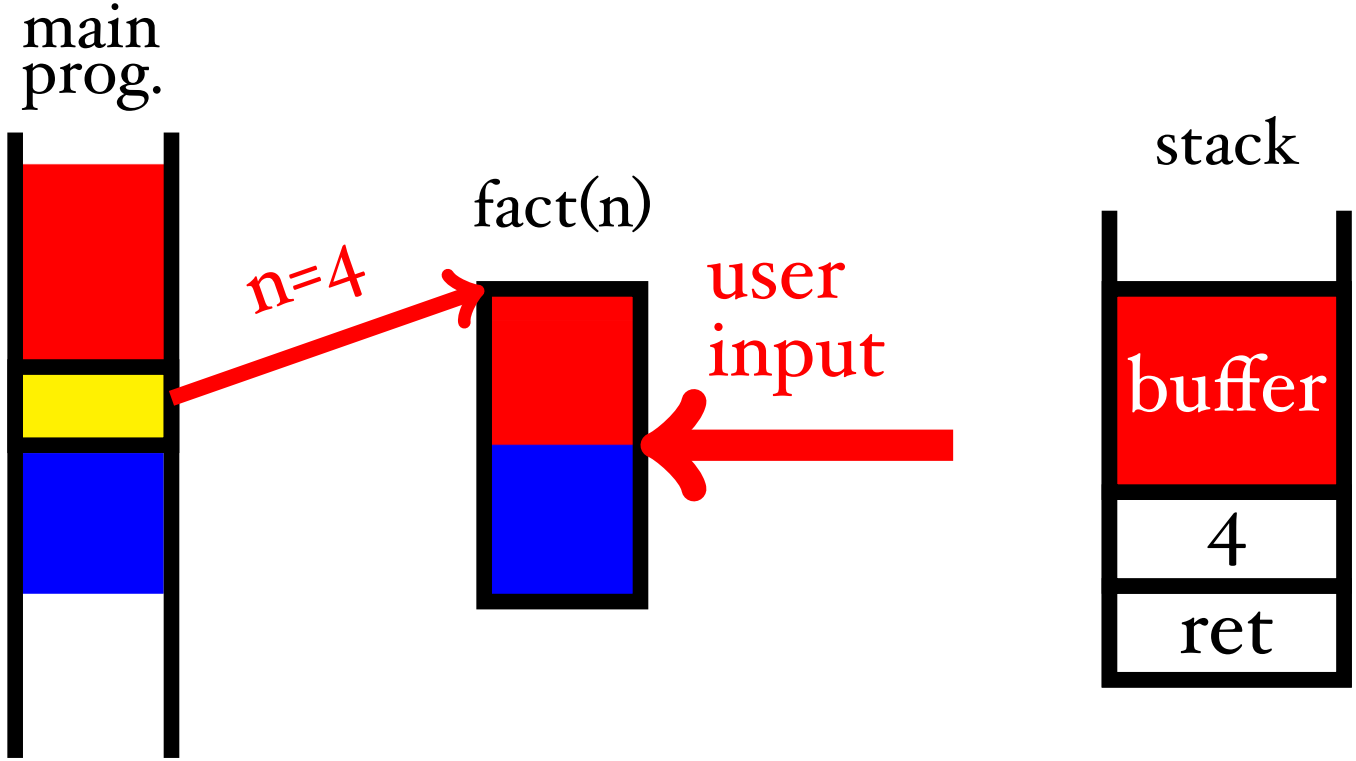

main
prog.
fact(n)
stack

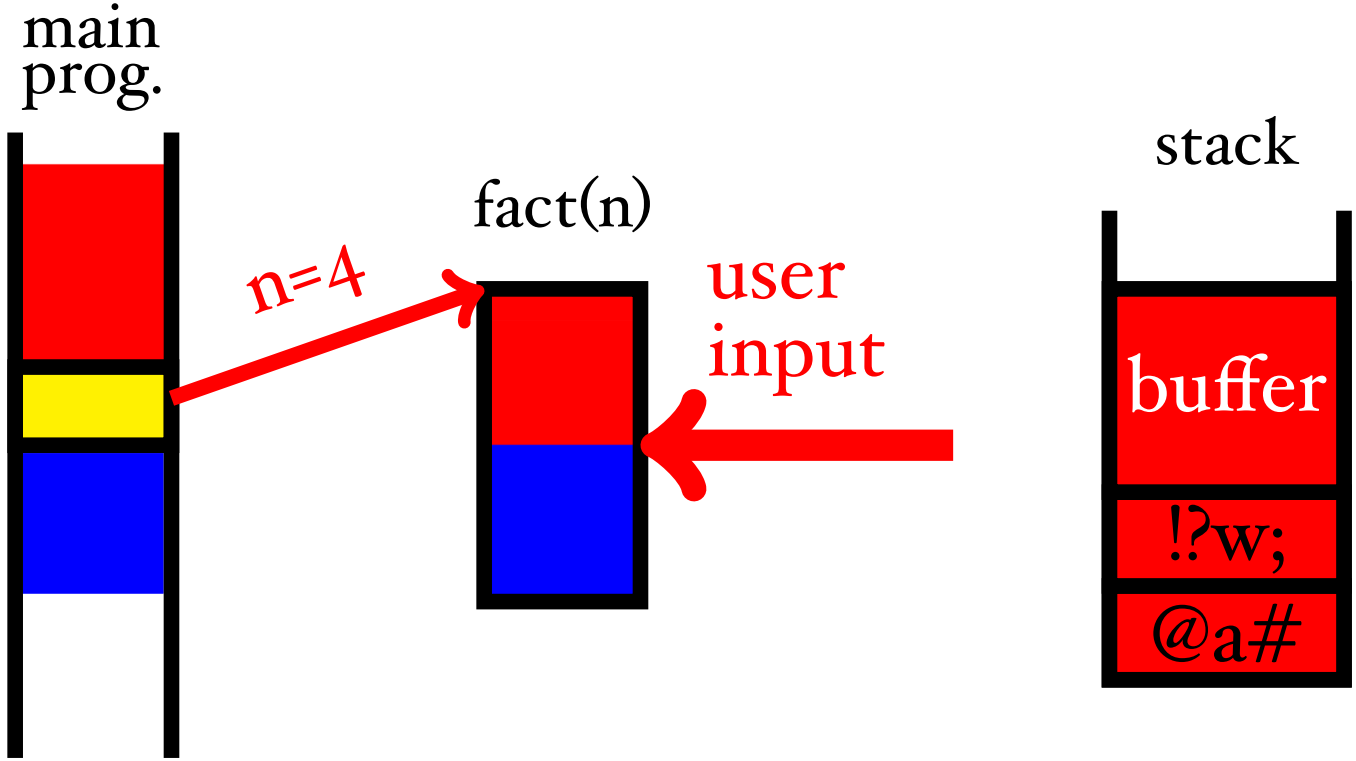

main
prog.
fact(n)
n=4

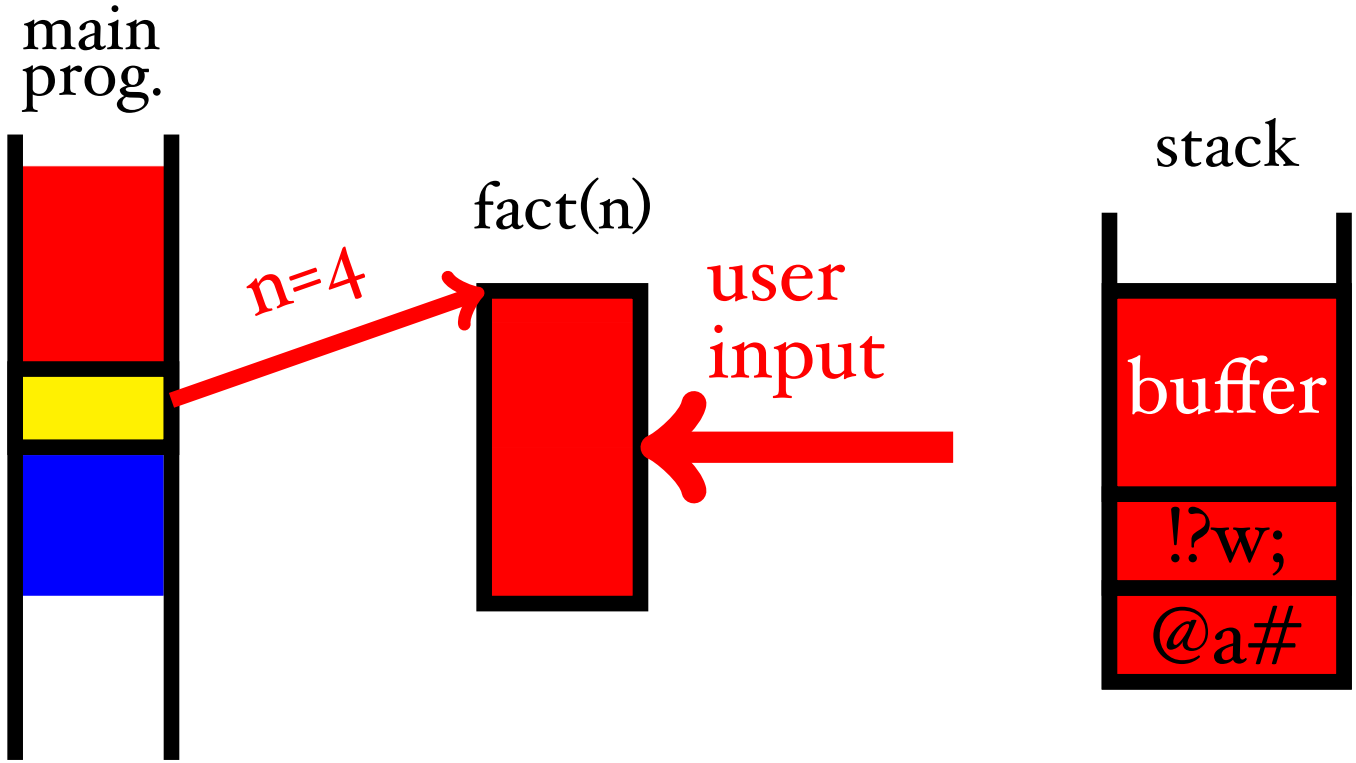

stack
4
ret

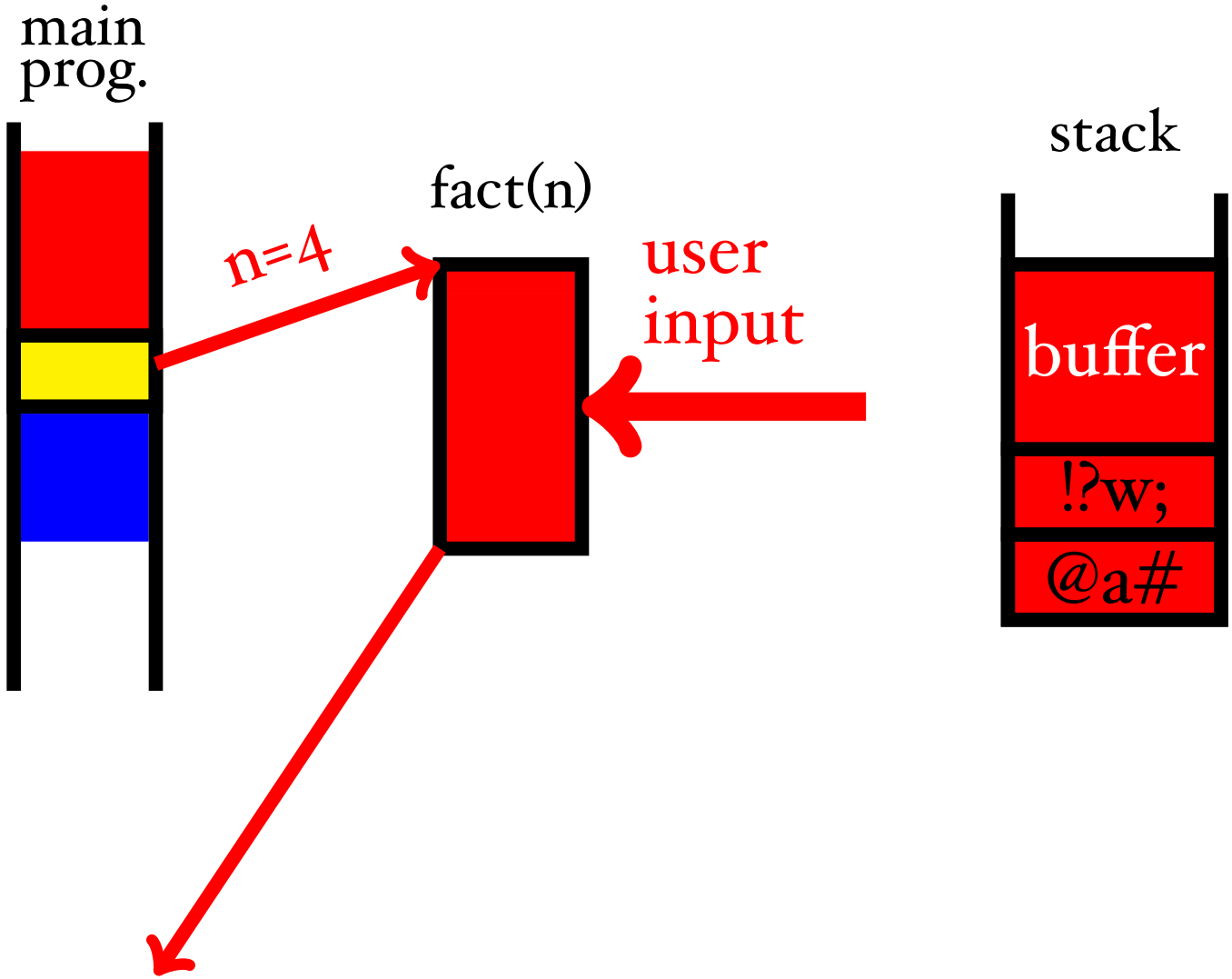

main
prog.
fact(n)
n=4


stack
buffer
4
ret


main
prog.
n=4
fact(n)
user
input
stack
buffer
4
ret

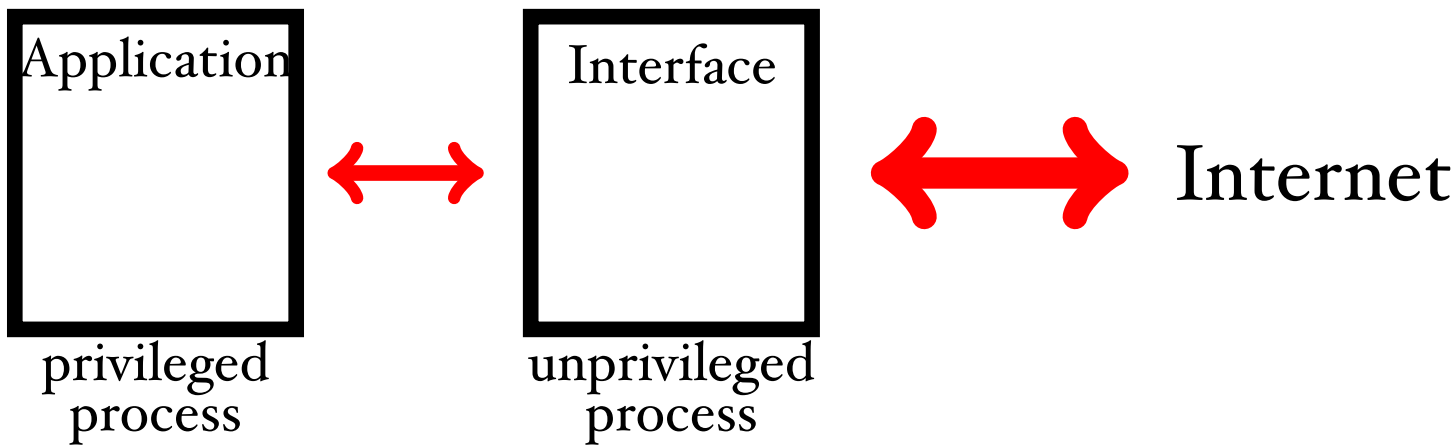

main
prog.
n=4
fact(n)
user
input
stack
buffer
!?w;
@a#


main
prog.
n=4
fact(n)
user
input
stack
buffer
!?w;
@a#


main
prog.
n=4
fact(n)
user
input
stack
buffer
!?w;
@a#

3rd Lecture: Unix Access Control

- privileges are specified by file access permissions ("everything is a file")



- the idea is make the attack surface smaller and mitigate the consequences of an attack

3rd Lecture: Unix Access Control

- when a file with setuid is executed, the resulting process will assume the UID given to the owner of the file

```
$ ls -ld . * */*
drwxr-xr-x 1 ping staff  32768 Apr  2 2010 .
-rw----r-- 1 ping students  31359 Jul 24 2011 manual.t
-r--rw--w- 1 bob students   4359 Jul 24 2011 report.tx
-rwsr--r-x 1 bob students 141359 Jun  1 2013 microedit
dr--r-xr-x 1 bob staff  32768 Jul 23 2011 src
-rw-r--r-- 1 bob staff  81359 Feb 28 2012 src/code.c
-r--rw---- 1 emma students    959 Jan 23 2012 src/code
```

8th Lecture: Privacy

- differential privacy for annonymizing research data

User	tell me $f(x) \Rightarrow$	Database
	$\Leftarrow f(x) + \text{noise}$	$x_1, \ldots, x_n$

- $f(x)$ can be released, if f is insensitive to individual entries $x_1, \ldots, x_n$
- The intuition: whatever is learned from the dataset would be learned regardless of whether x_i participates

8th Lecture: Privacy

- differential privacy for annonymizing research data

User	tell me $f(x)$ $\Rightarrow$	Database
	$\Leftarrow f(x) + \text{noise}$	$x_1, \ldots, x_n$

- $f(x)$ can be released, if f is insensitive to individual entries $x_1, \ldots, x_n$
- The intuition: whatever is learned from the dataset would be learned regardless of whether x_i participates

- Tor webservice

9th Lecture: Privacy

- zero-knowledge proofs
- requires NP problems, for example graph isomorphisms

9th Lecture: Privacy

- zero-knowledge proofs
- requires NP problems, for example graph isomorphisms

- random number generators