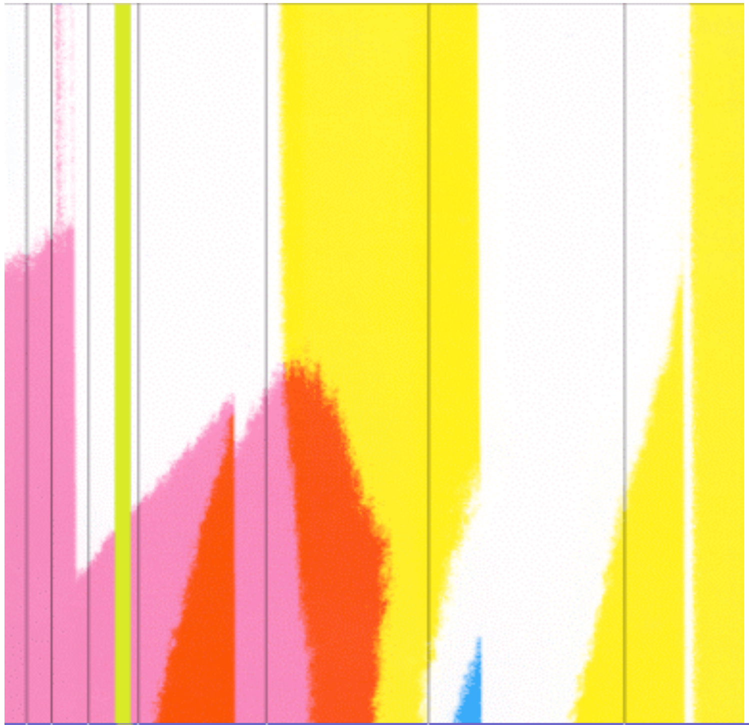


Arachne:
A Dynamic Weaver for legacy C Applications

16 Décembre 2008

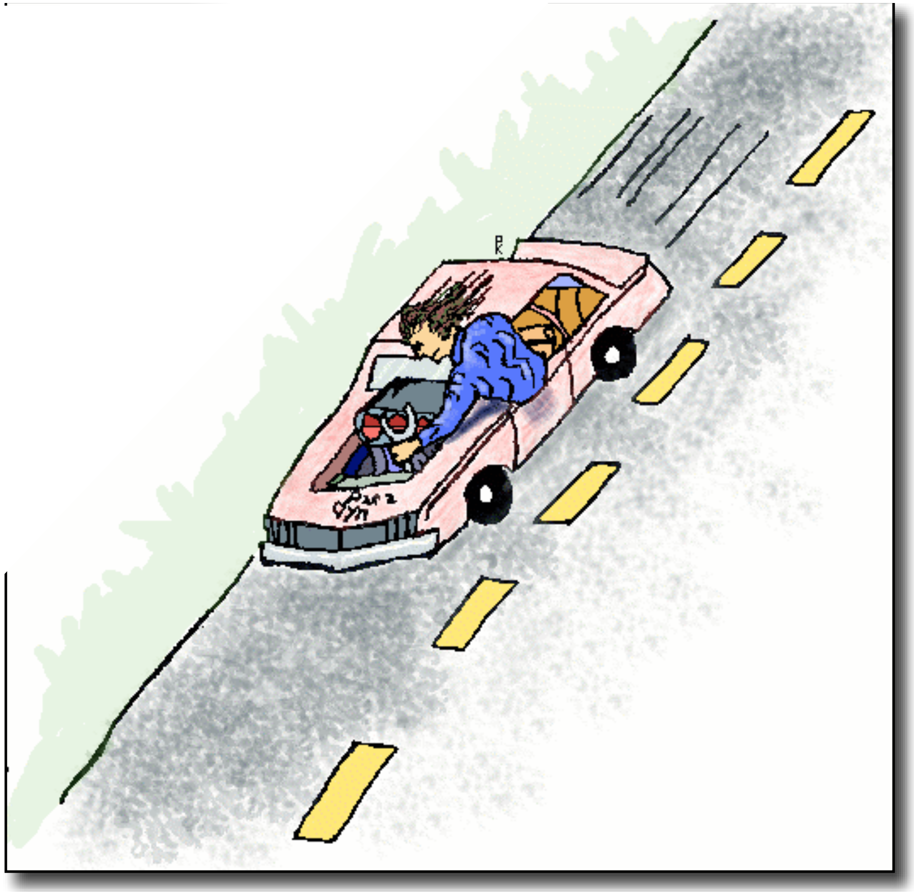
INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

Jean-Marc Menaud
Ascola EMN/INRIA - LINA

Journée Logiciel embarqué,
Rennes - Bretagne Atlantique
Pole Images & réseaux le 16 Décembre 2008

Arachne



Dynamic Adaptation of Applications

Target Applications :

- Linux Kernel, Server web,
- Server ftp, demon, web cache...

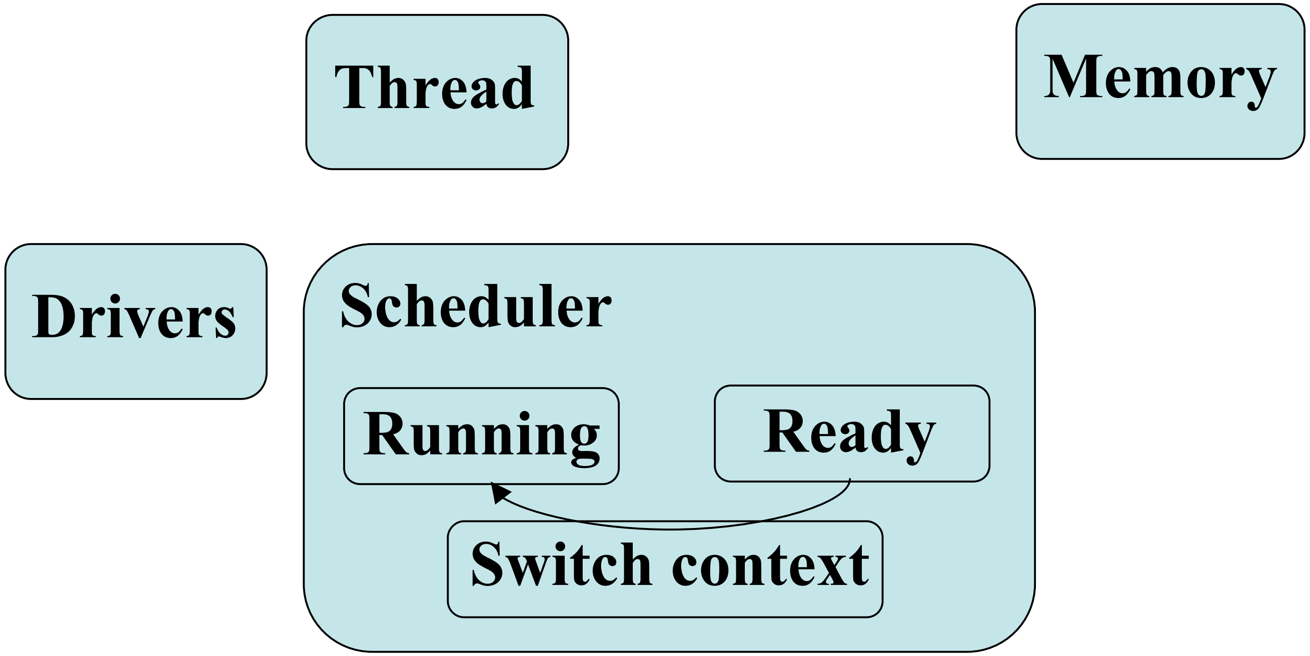
Our constraints :

- C legacy code,
- Critical Performances,
- Uninterruptible

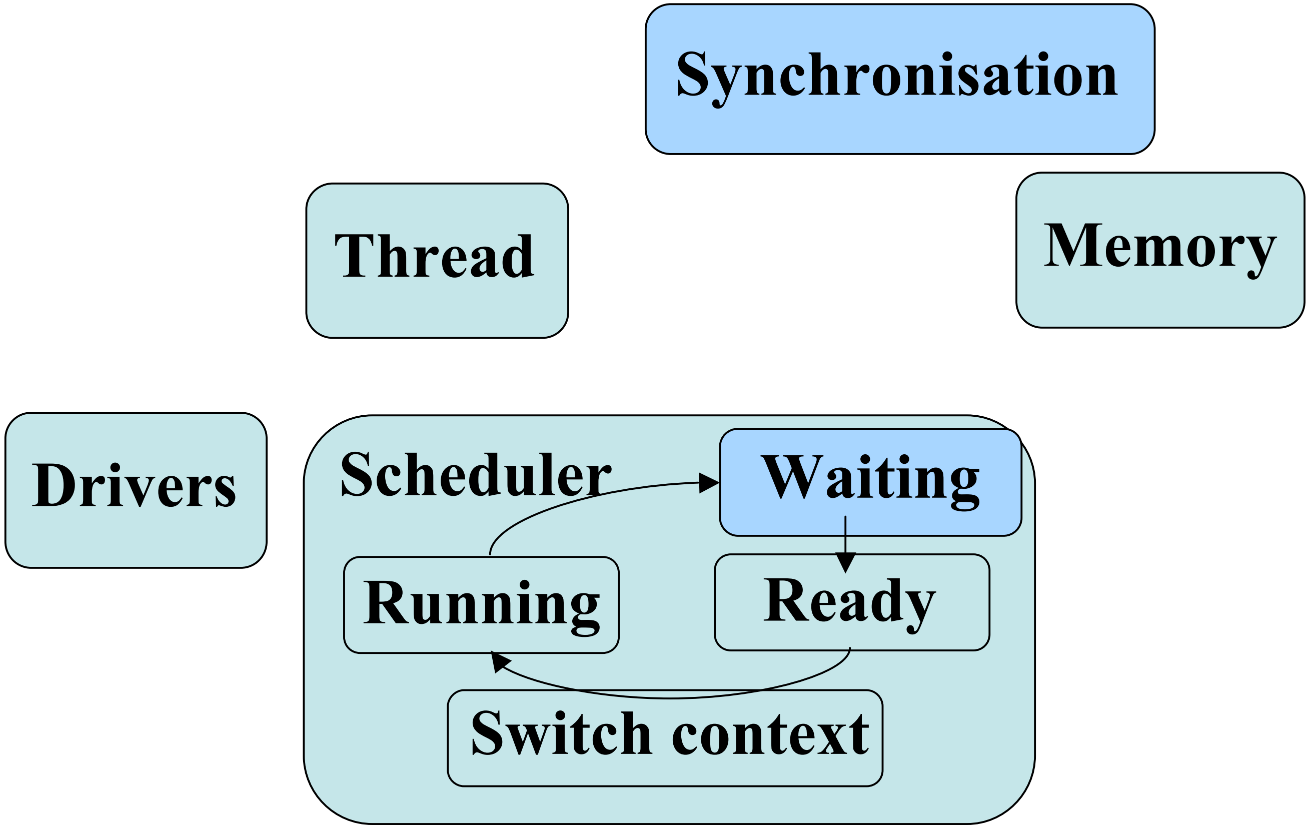
A Dynamic AOSD system for legacy applications written in C.

- Without source or binary code preparation
- Without service interruption
- Without performance loss
- By binary code rewriting

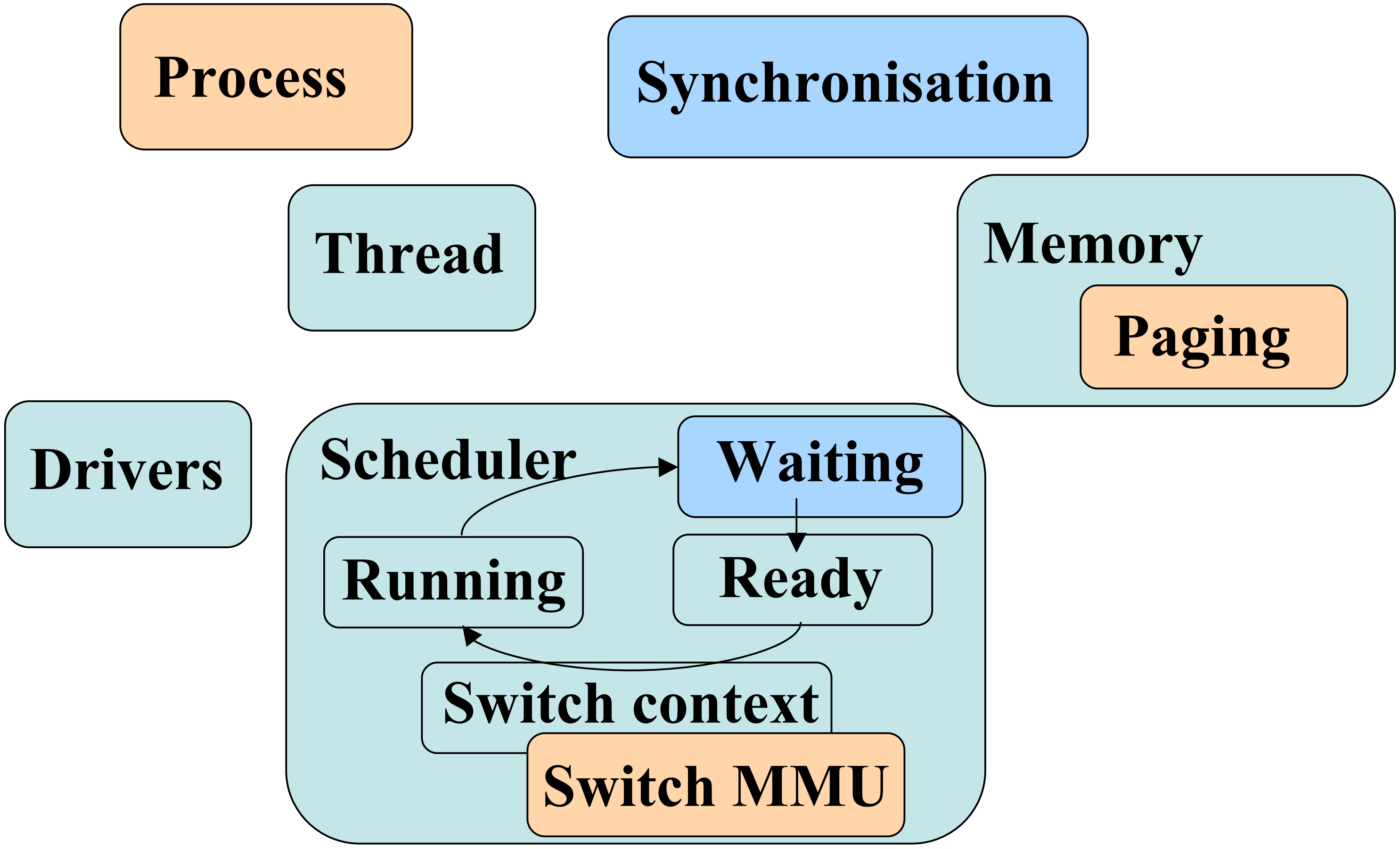
Os Construction



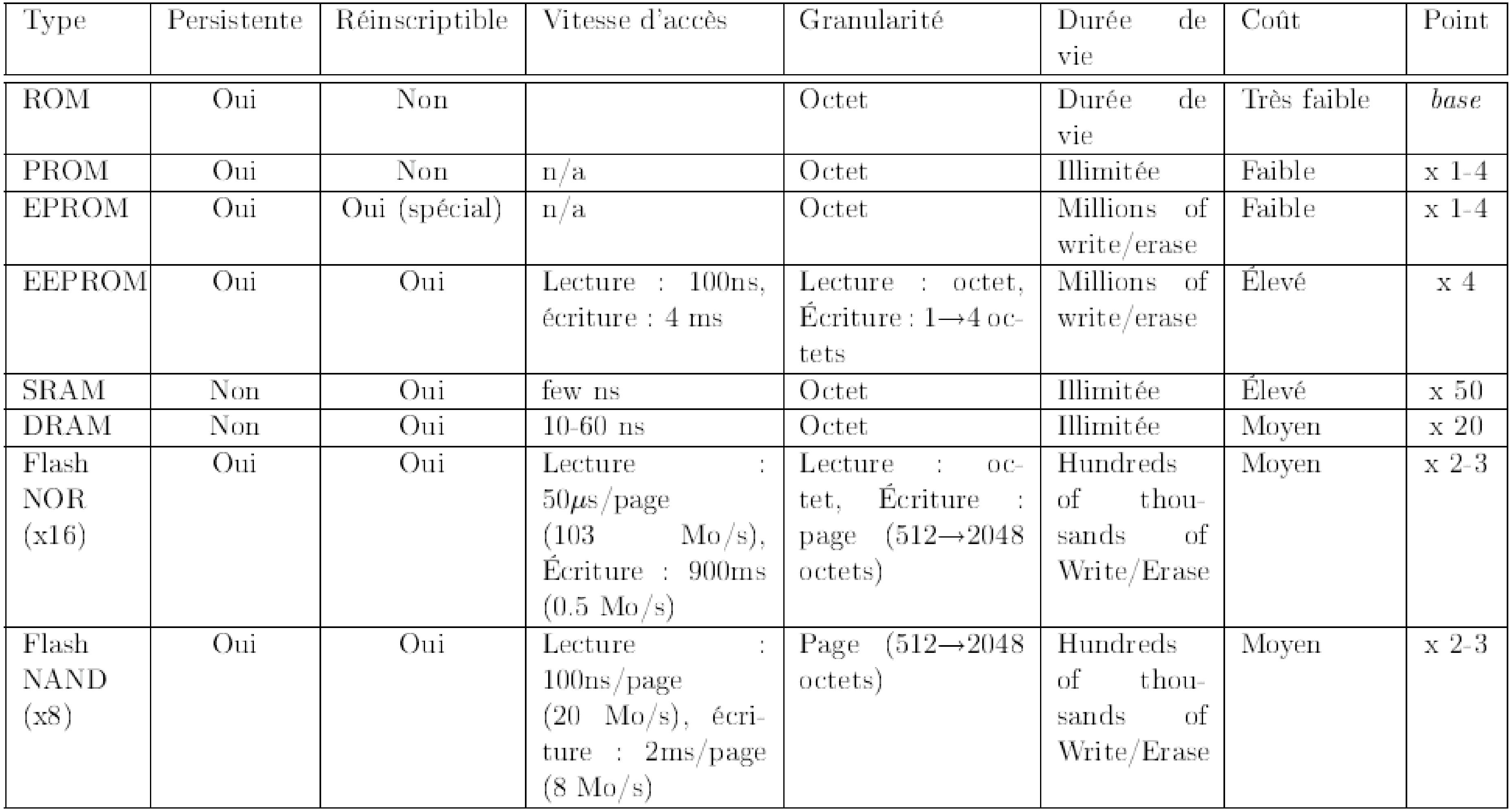
Os Construction



Os Construction



Memory Manager in Embedded System

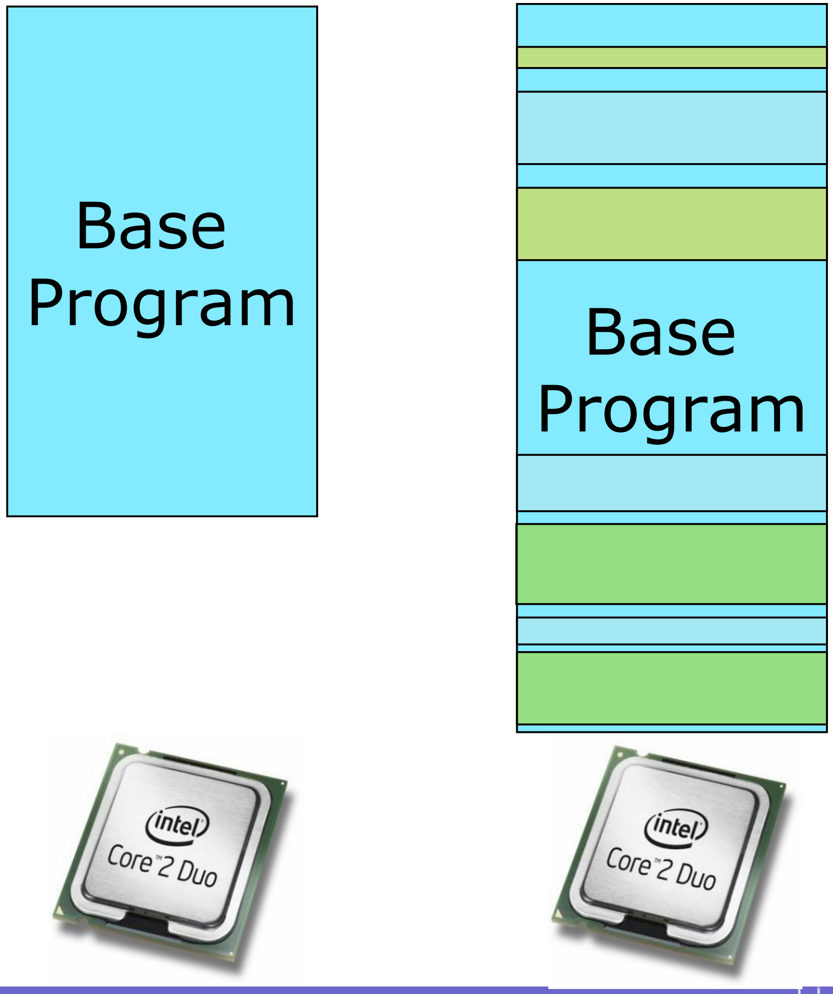


Type	Persistente	Réinscriptible	Vitesse d'accès	Granularité	Durée de vie	Coût	Point
ROM	Oui	Non		Octet	Durée de vie	Très faible	*base*
PROM	Oui	Non	n/a	Octet	Illimitée	Faible	x 1-4
EPROM	Oui	Oui (spécial)	n/a	Octet	Millions of write/erase	Faible	x 1-4
EEPROM	Oui	Oui	Lecture : 100ns, écriture : 4 ms	Lecture : octet, Écriture : 1→4 octets	Millions of write/erase	Élevé	x 4
SRAM	Non	Oui	few ns	Octet	Illimitée	Élevé	x 50
DRAM	Non	Oui	10-60 ns	Octet	Illimitée	Moyen	x 20
Flash NOR (x16)	Oui	Oui	Lecture : 50μs/page (103 Mo/s), Écriture : 900ms (0.5 Mo/s)	Lecture : octet, Écriture : page (512→2048 octets)	Hundreds of thousands of Write/Erase	Moyen	x 2-3
Flash NAND (x8)	Oui	Oui	Lecture : 100ns/page (20 Mo/s), écriture : 2ms/page (8 Mo/s)	Page (512→2048 octets)	Hundreds of thousands of Write/Erase	Moyen	x 2-3

AOP in few words

Base Program

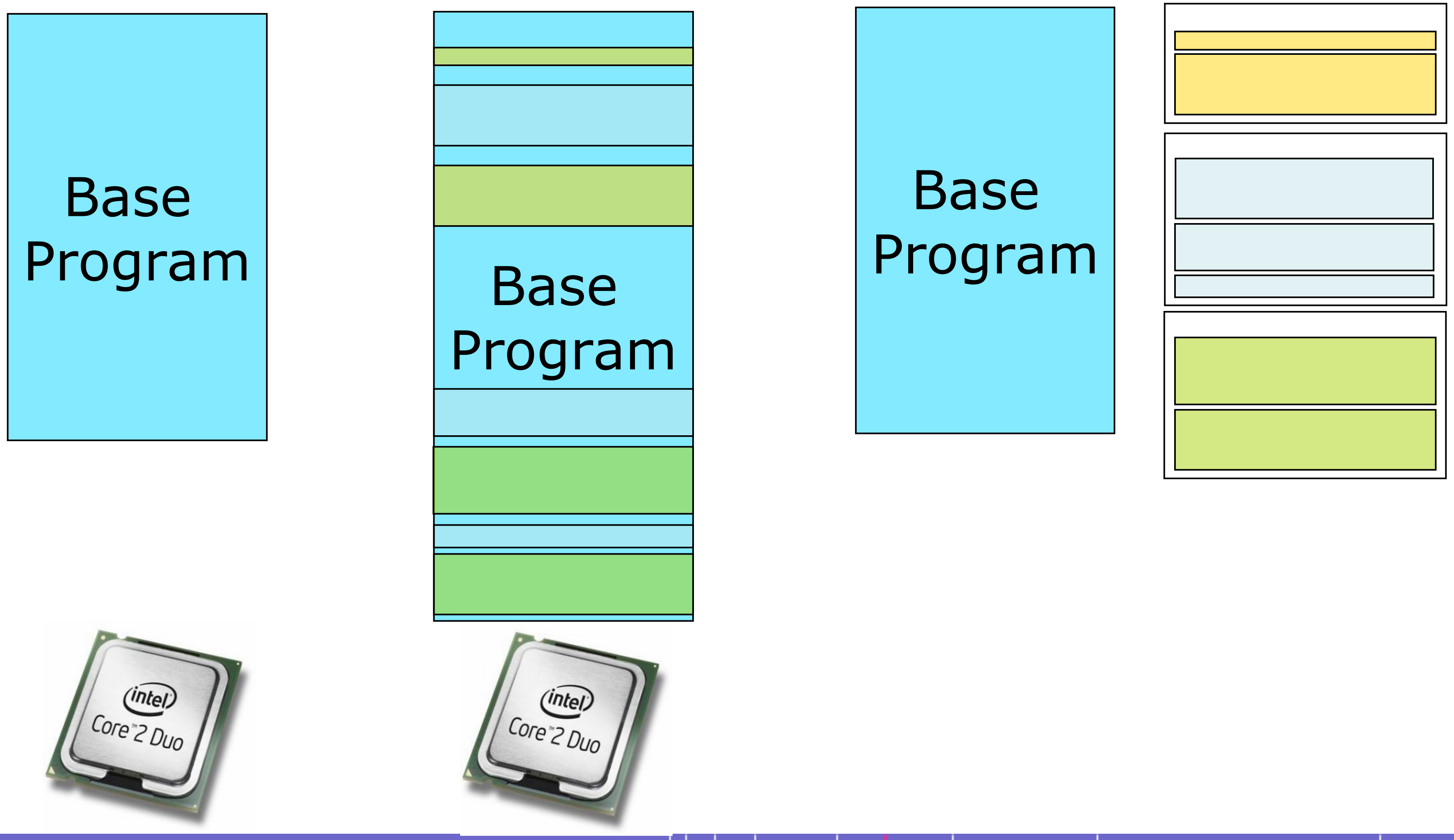
AOP in few words



Base Program

Base Program

AOP in few words

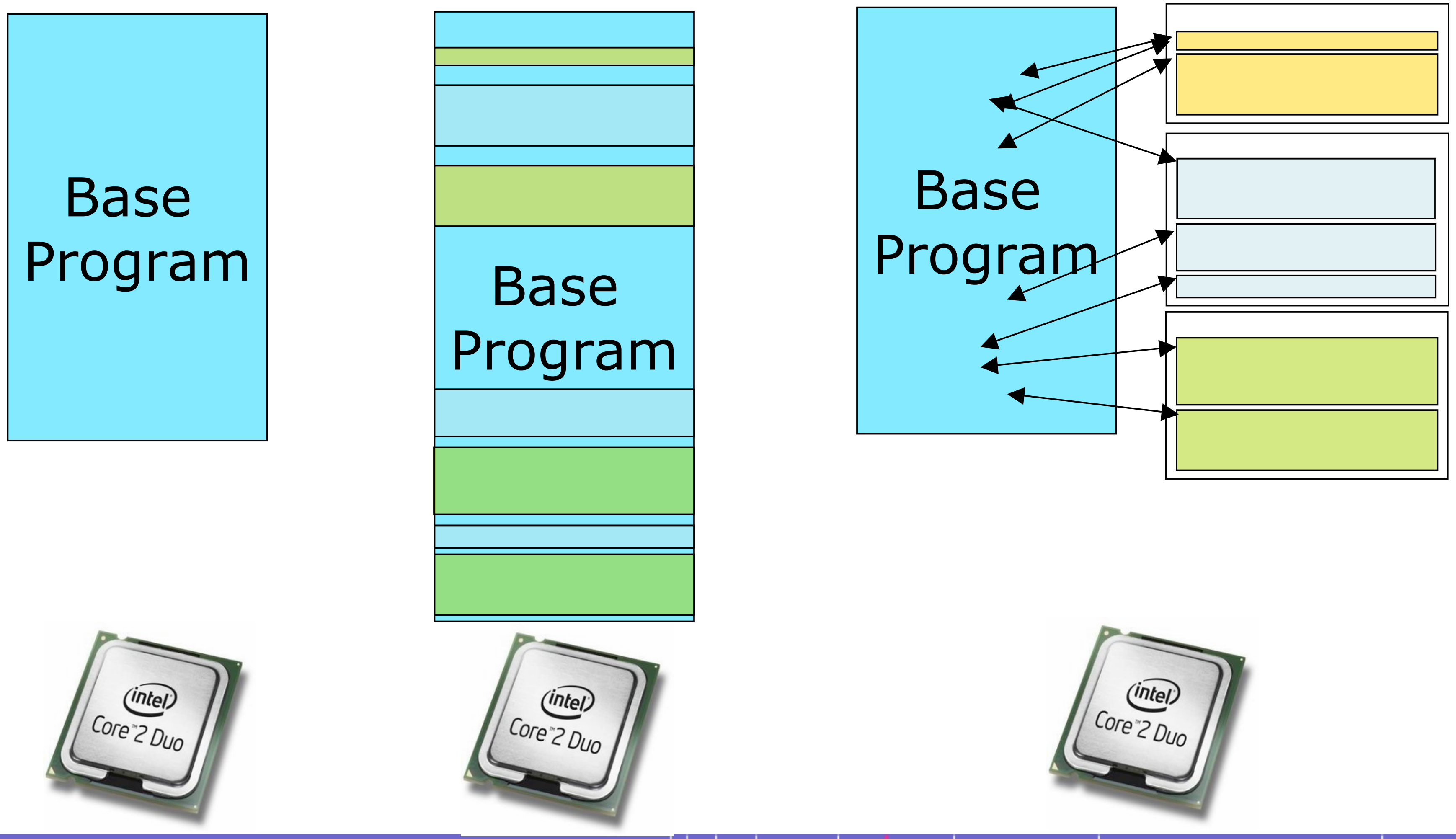


Base Program

Base Program

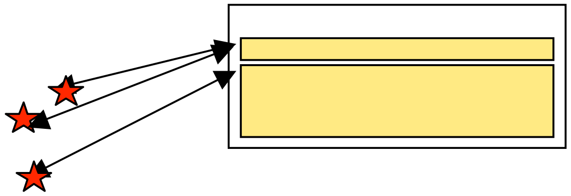
Base Program

AOP in few words



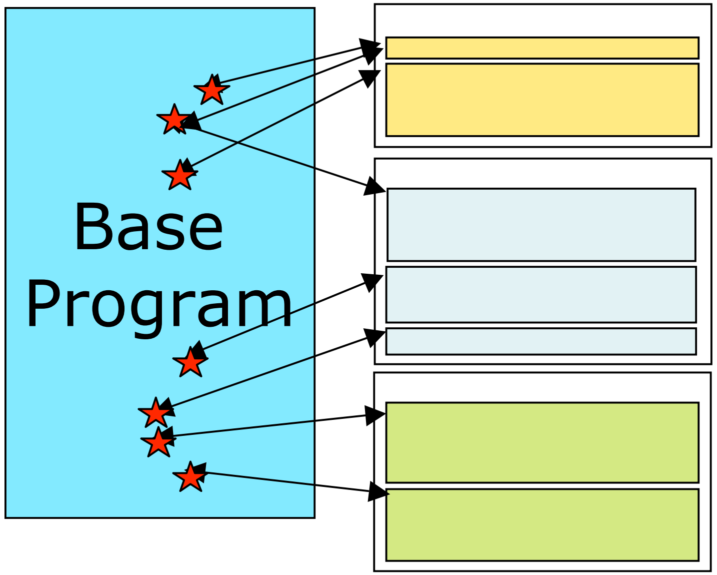
Arachne : EAOP Transposition for C

An Aspect



An Aspect with Arachne

- Joinpoint ★
  - Global variable read/write access
  - Function call
- Pointcut ↔
  - Logical operator
  - cflow à la AspectJ and seq
- Advice ▭
  - C and proceed



A concret exemple

```
...
int *x ;
x = (int *)malloc(sizeof(int) * 4);
if (x == NULL) {
      /* rountine to handle the case */
      /* when memory allocation failed */
}
      /* routine for handling the normal case */
...
```

AOP Solution

```
after(void *s):call($ malloc(...)) && result(s) {
     if ((char *)(s) == NULL) {
     /* rountine to handle the case */
     /* when memory allocation failed */
     }}
```

```
...
int *x ;
x = (int *)malloc(sizeof(int) * 4);
/* routine for handling the normal case */
...
```

Arachne : Dynamic weaving

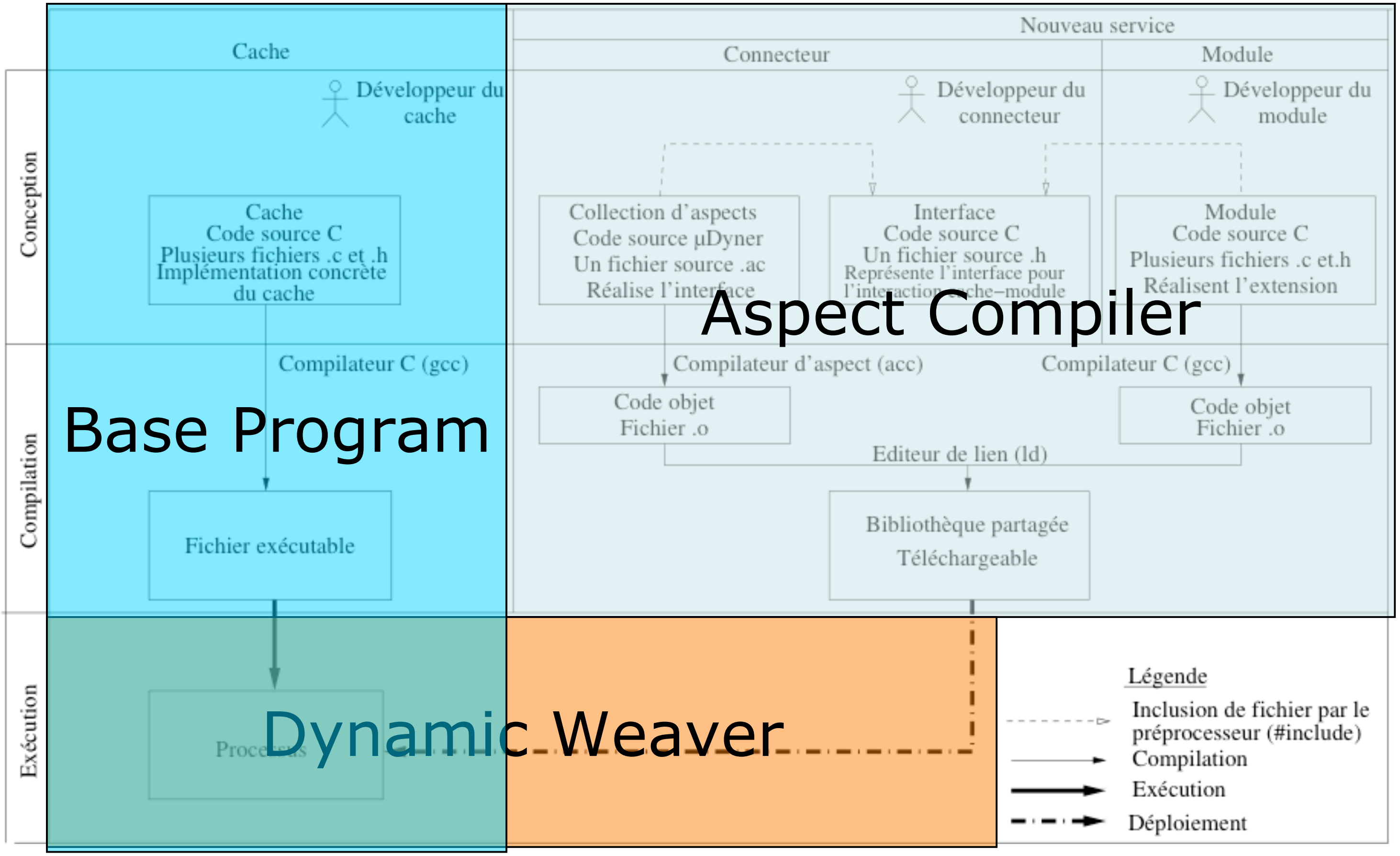
Pro

- Rebooting the kernel, which results in downtime and loss of state (e.g., all network connections)
- Can be activated and deactivated without the need to recompile
- Small amount of memory will be expended to store the replacement code

Cons

- Does not make semantic changes to the software persistent data structures

Aspect Life cycle



3 Phases

🟦 Conception / Deployment / Execution

⬜ Aspect Development

🟧 Aspect weaving

BankAccount.c

```
int activity;

iint deposit(char *name, int val) {
          my_stream = fopen (name, "a");
          fprintf (my_stream, "+%d\n",val);
          fclose (my_stream);
          activity += val;
}

int withdraw(char *name,int val){
          my_stream = fopen (name, "a");
          fprintf (my_stream, "-%d\n",val);
          fclose (my_stream);
          activity += val;
}


int transfert(char *from, char *to, int val) {
          withdraw(from,val);
          deposit(to,val);
}

int balance(char *name){
          int temp=0;
          my_stream = fopen (name, "r");

          while ( fscanf(my_stream,"%s\n",buf)>0) {
                    temp+=atoi(buf);
          }

          fclose (my_stream);
          return temp;
}
```

First Aspect :From Francs to Euros with tax

```
int activity;

iint deposit(char *name, int val) {
                my_stream = fopen (name, "a");
                fprintf (my_stream, "+%d\n",val);
                fclose (my_stream);
                activity += val;
}

int withdraw(char *name,int val){
                my_stream = fopen (name, "a");
                fprintf (my_stream, "-%d\n",val);
                fclose (my_stream);
                activity += val;
}


int transfert(char *from, char *to, int val) {
                withdraw(from,val);
                deposit(to,val);
}

int balance(char *name){
                int temp=0;
                my_stream = fopen (name, "r");

                while ( fscanf(my_stream,"%s\n",buf)>0) {
                                temp+=atoi(buf);
                }

                fclose (my_stream);
                return temp;
}
```

```
#include <arachne/aspect.h>

#define tax 10


aspect Activity ::
                writeglobal (activity) && value(k)
                        then {
                                activity = k*6.56;
                                }

aspect Activity::
                readglobal (activite)
                        then {
                                return activity/6.56;
                                }

aspect tax_deposit ::
                call (int deposit(char *name, int val))
)
                        then {
                                proceed( "TAX", tax);
                                proceed(name, val - tax);
                                return 0;
                                }
```

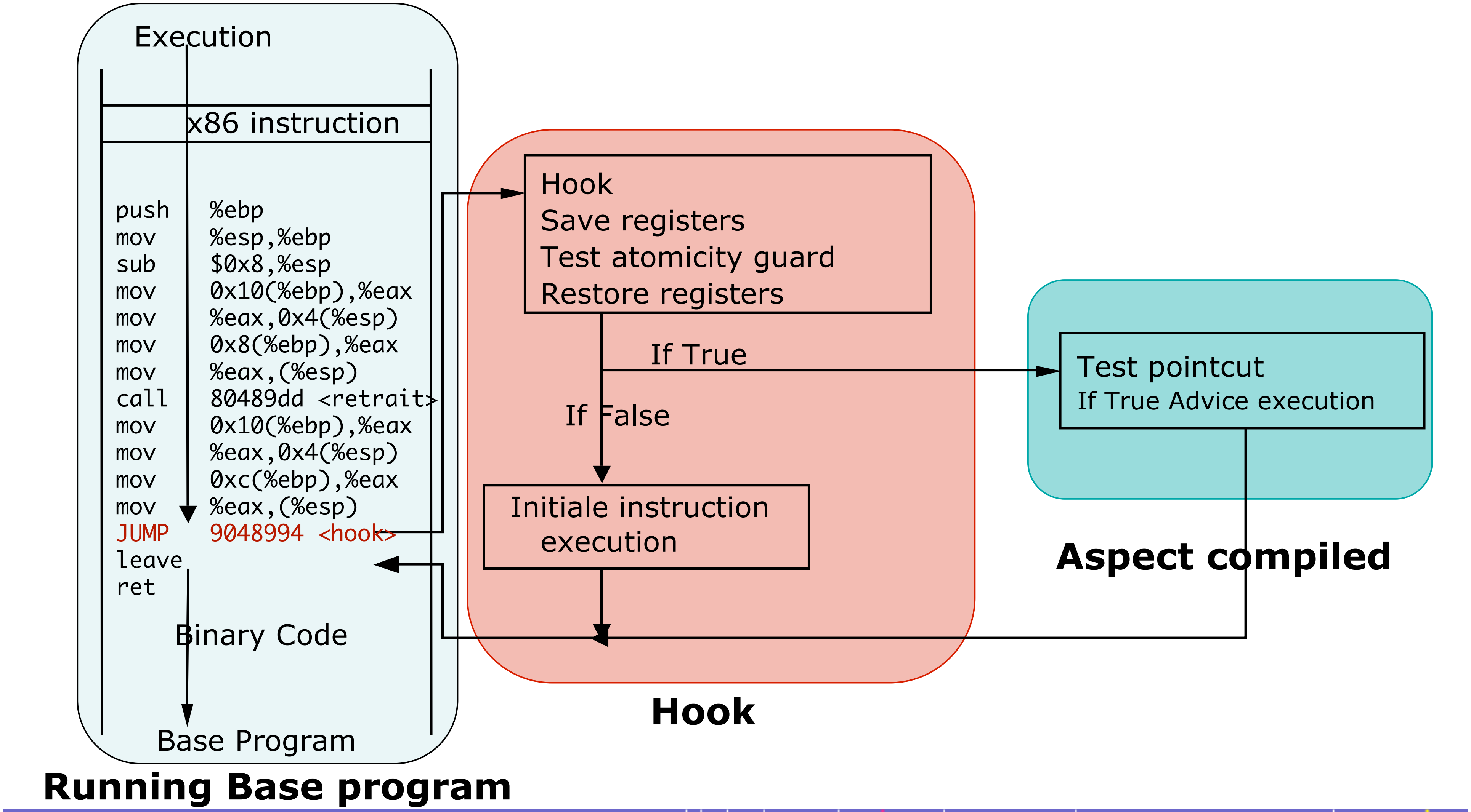
BankAccount

Relation between source code and binary code

```
iint deposit(char *name, int val) {
		my_stream = fopen (name, "a");
		fprintf (my_stream, "+%d\n",val);
		fclose (my_stream);
		activity += val;
}

int withdraw(char *name,int val){
		my_stream = fopen (name, "a");
		fprintf (my_stream, "-%d\n",val);
		fclose (my_stream);
		activity += val;
}


int transfert(char *from, char *to, int val) {
		withdraw(from,val);
		deposit(to,val);
}

int balance(char *name){
		int temp=0;
		my_stream = fopen (name, "r");

		while ( fscanf(my_stream,"%s\n",buf)>0) {
				temp+=atoi(buf);
		}

		fclose (my_stream);
		return temp;
}
```

```
08048a95 <transfert>:
 8048a95:       55                      push   %ebp
 8048a96:       89 e5                   mov    %esp,%ebp
 8048a98:       83 ec 08                sub    $0x8,%esp
 8048a9b:       8b 45 10                mov    0x10(%ebp),%eax
 8048a9e:       89 44 24 04             mov    %eax,0x4(%esp)
 8048aa2:       8b 45 08                mov    0x8(%ebp),%eax
 8048aa5:       89 04 24                mov    %eax,(%esp)
 8048aa8:       e8 30 ff ff ff          call   80489dd <withdraw>
 8048aad:       8b 45 10                mov    0x10(%ebp),%eax
 8048ab0:       89 44 24 04             mov    %eax,0x4(%esp)
 8048ab4:       8b 45 0c                mov    0xc(%ebp),%eax
 8048ab7:       89 04 24                mov    %eax,(%esp)
 8048aba:       e8 d5 fe ff ff          call   8048994 <deposit>
 8048abf:       c9                      leave
 8048ac0:       c3                      ret
```

Weaving



Execution

x86 instruction

```
push    %ebp
mov     %esp,%ebp
sub     $0x8,%esp
mov     0x10(%ebp),%eax
mov     %eax,0x4(%esp)
mov     0x8(%ebp),%eax
mov     %eax,(%esp)
call    80489dd <retrait>
mov     0x10(%ebp),%eax
mov     %eax,0x4(%esp)
mov     0xc(%ebp),%eax
mov     %eax,(%esp)
JUMP    9048994 <hook>
leave
ret
```

Binary Code

Base Program

Running Base program

Hook
Save registers
Test atomicity guard
Restore registers

If True

If False

Initiale instruction
execution

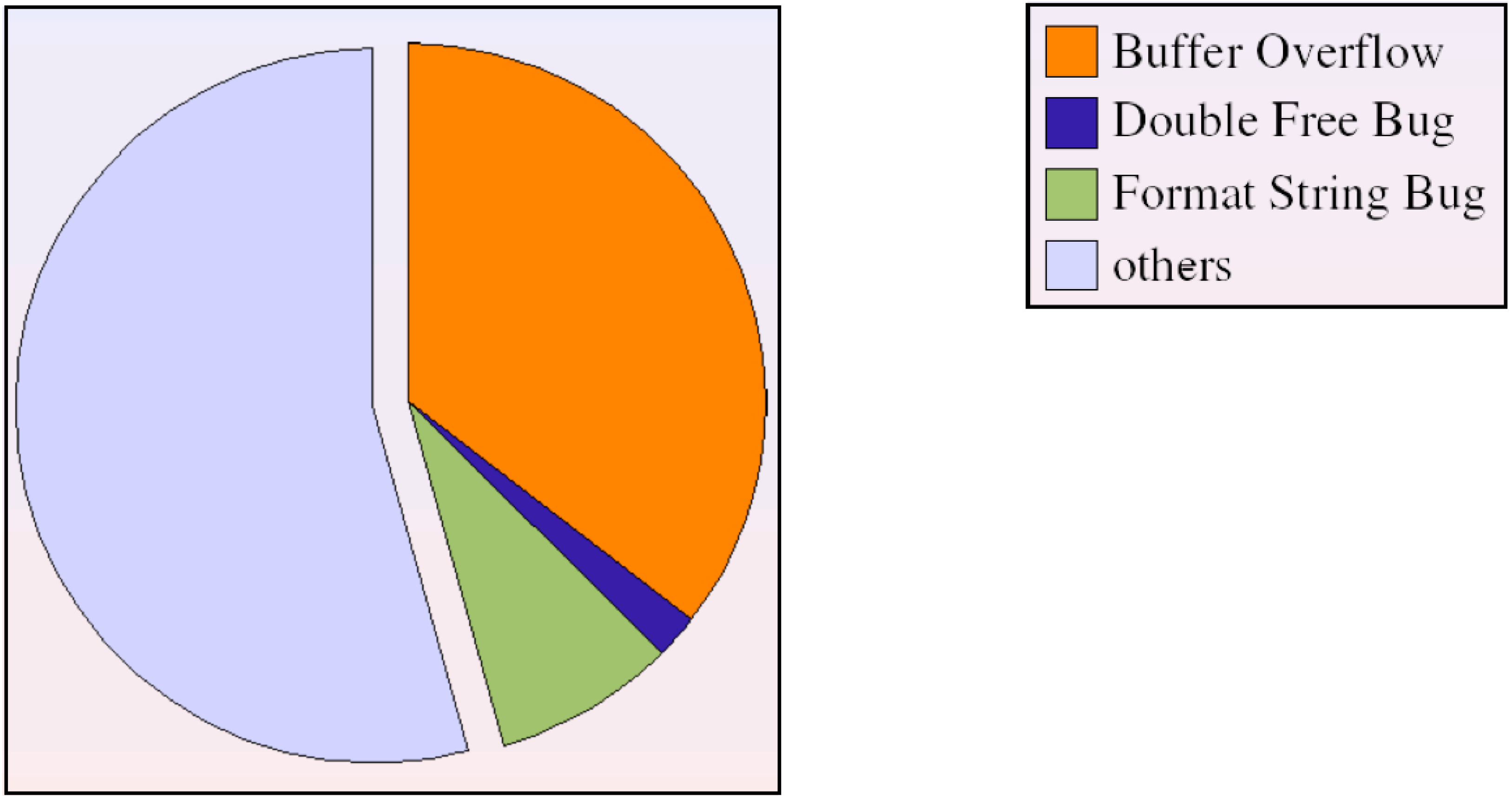
Hook

Test pointcut
If True Advice execution

Aspect compiled

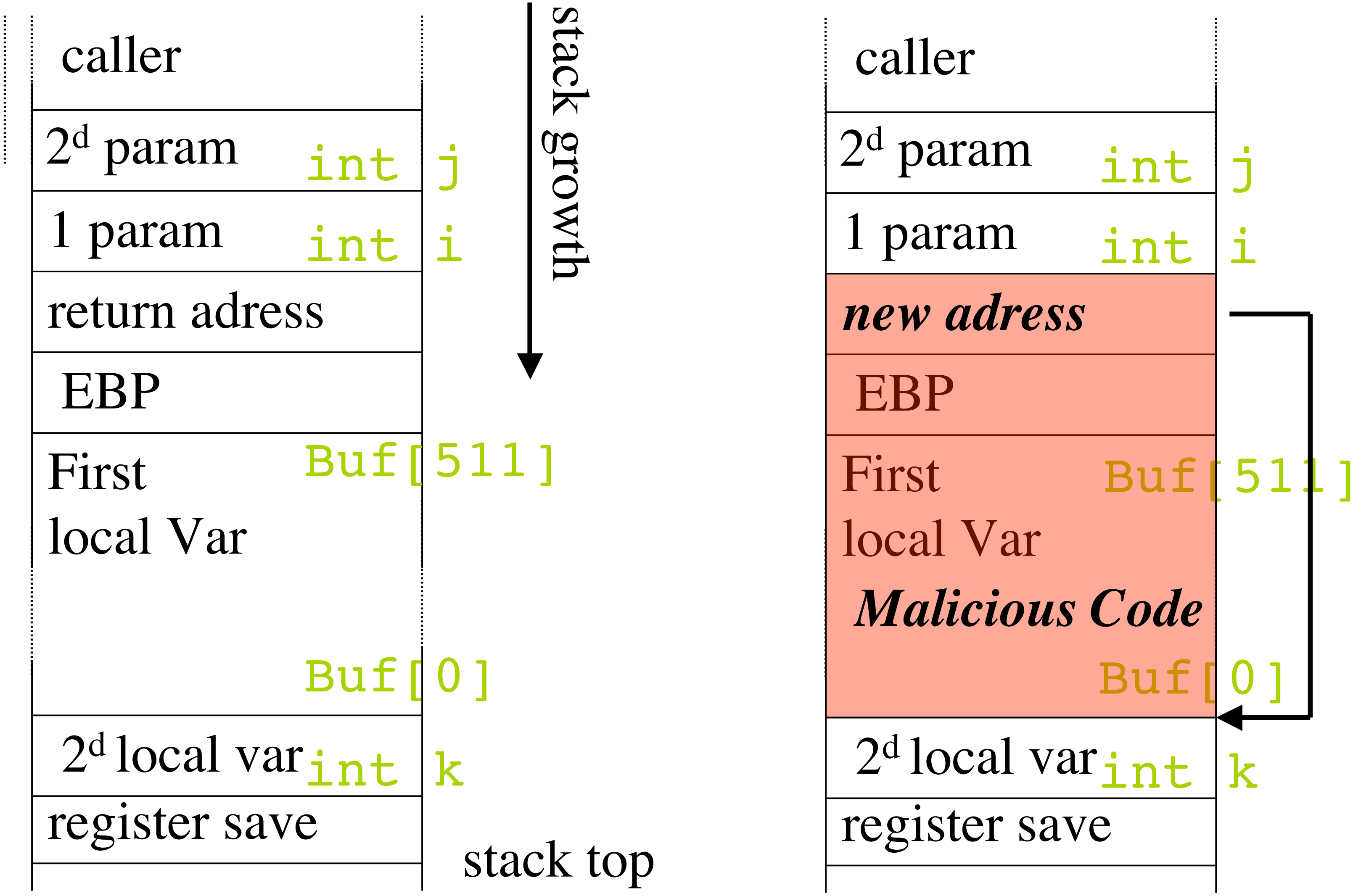
Application ?

C-related Bugs



Buffer overflow forensic

```
void func(int i, int j) {
char buf[512];
int k;
```



caller

2d param int j

1 param int i

return adress

EBP

First local Var Buf[511]

Buf[0]

2d local var int k

register save

stack growth

stack top

caller

2d param int j

1 param int i

new adress

EBP

First local Var Buf[511]

Malicious Code

Buf[0]

2d local var int k

register save

```
static void
ftpStateFree(int fdnotused, void *data)
{
    FtpStateData *ftpState = data;
    if (ftpState == NULL)
            return;
    debug(9, 3) ("ftpStateFree: %s\n", storeUrl(ftpState->entry));
    storeUnregisterAbort(ftpState->entry);
    storeUnlockObject(ftpState->entry);
    if (ftpState->reply_hdr) {
            memFree(ftpState->reply_hdr, MEM_8K_BUF);
            /* this seems unnecessary, but people report SEGV's
             * when freeing memory in this function */
            ftpState->reply_hdr = NULL;
    }
    requestUnlink(ftpState->request);
    if (ftpState->ctrl.buf) {
            ftpState->ctrl.freefunc(ftpState->ctrl.buf);
            /* this seems unnecessary, but people report SEGV's
             * when freeing memory in this function */
            ftpState->ctrl.buf = NULL;
    }
    if (ftpState->data.buf) {
            ftpState->data.freefunc(ftpState->data.buf);
            /* this seems unnecessary, but people report SEGV's
             * when freeing memory in this function */
            ftpState->data.buf = NULL;
    }
    if (ftpState->pathcomps)
            wordlistDestroy(&ftpState->pathcomps);
    if (ftpState->ctrl.message)
            wordlistDestroy(&ftpState->ctrl.message);
    if (ftpState->cwd_message)
            wordlistDestroy(&ftpState->cwd_message);
    safe_free(ftpState->ctrl.last_reply);
    safe_free(ftpState->ctrl.last_command);
    safe_free(ftpState->old_request);
    safe_free(ftpState->old_reply);
    safe_free(ftpState->old_filepath);
    stringClean(&ftpState->title_url);
    stringClean(&ftpState->base_href);
    safe_free(ftpState->filepath);
    safe_free(ftpState->data.host);
    if (ftpState->data.fd > -1) {
            comm_close(ftpState->data.fd);
            ftpState->data.fd = -1;
    }
    cbdataFree(ftpState);
}
```

```
static void
ftpListingStart(FtpStateData * ftpState)
{
    StoreEntry *e = ftpState->entry;
    wordlist *w;
    char *dirup;
    int i, j, k;
    char *title;
    storeBuffer(e);
    storeAppendPrintf(e, "<!-- HTML listing generated by Squid %s -->\n",
            version_string);
    storeAppendPrintf(e, "<!-- %s -->\n", mkrfc1123(squid_curtime));
    storeAppendPrintf(e, "<HTML><HEAD><TITLE>\n");
    storeAppendPrintf(e, "FTP Directory: %s\n",
            html_quote(strBuf(ftpState->title_url)));
    storeAppendPrintf(e, "</TITLE>\n");
    if (ftpState->flags.use_base)
            storeAppendPrintf(e, "<BASE HREF=\"%s\">\n",
                html_quote(strBuf(ftpState->base_href)));
    storeAppendPrintf(e, "</HEAD><BODY>\n");
    if (ftpState->cwd_message) {
            storeAppendPrintf(e, "<PRE>\n");
            for (w = ftpState->cwd_message; w; w = w->next)
                storeAppendPrintf(e, "%s\n", html_quote(w->key));
            storeAppendPrintf(e, "</PRE>\n");
            storeAppendPrintf(e, "<HR>\n");
            wordlistDestroy(&ftpState->cwd_message);
    }
    storeAppendPrintf(e, "<H2>\n");
    storeAppendPrintf(e, "FTP Directory: ");
    /* "ftp://" == 6 characters */
    assert(strLen(ftpState->title_url) >= 6);
    title = html_quote(strBuf(ftpState->title_url));
    for (i = 6, j = 0; title[i]; j = i) {
            storeAppendPrintf(e, "<A HREF=\"");
            i += strcspn(&title[i], "/");
            if (title[i] == '/')
                i++;
            for (k = 0; k < i; k++)
                storeAppendPrintf(e, "%c", title[k]);
            storeAppendPrintf(e, "\">");
            for (k = j; k < i - 1; k++)
                storeAppendPrintf(e, "%c", title[k]);
            if (strBuf(ftpState->title_url)[k] != '/')
                storeAppendPrintf(e, "%c", title[k++]);
            storeAppendPrintf(e, "</A>");
            if (k < i)
                storeAppendPrintf(e, "%c", title[k++]);
            if (i == j) {
                /* Error guard, or "assert" */
                storeAppendPrintf(e, "ERROR: Failed to parse URL: %s\n",
                              html_quote(strBuf(ftpState->title_url)));
                debug(9, 0) ("Failed to parse URL:
                               %s\n", strBuf(ftpState->title_url));
                break;
```

Security Aspect

[PRDC 2005]

```
int depot(char *nom, int val) {...}

int retrait(char *nom,int val){...}

int solde(char *nom){...}

int transfert(char *de, char *vers, int val) {...}

int parserRequete(char *req){
// Analyse de la requête
// appel aux fonctions appropriées
}

int sauvegarderRequete(char *t) {
  char  in[256], *p=in,i;

  do { read(0,p,1); p++ ;}
     while( (*(p-1)) != '\n');
  *(p-1)='\0';

  memcpy(t,in,256);
  return 1;
}

void traiterRequetes(){
  while (sauvegarderRequete(tampon)==1)
                          parserRequete(tampon);}

int main (int argc, char** argv ) {
...
  while(true){ /* gere les connexions clients*/
    client = accept(serveur,NULL, 0);
    traiterRequetes();
    close(client);
  }
}
```

```
aspect secure ::
          call (int sauvegarderRequete(char *t) ) then {
            char  in[256], *p=in,i;

            do { read(0,p,1); p++ ;}
               while( ((*(p-1)) != '\n') && (p<(in+254)) );;
            *(p-1)='\0';


            if(p==(in+254)) do { read(0,&c,1) ; }
                        while (( c!='\0') && (c!= '\n'));

            memcpy(t,in,256);
            return 1;
}
```

[AOSD 2003]

An aspect « Transfert tax»

The control flow

```
iint deposit(char *name, int val) {
        my_stream = fopen (name, "a");
        fprintf (my_stream, "+%d\n",val);
        fclose (my_stream);
        activity += val;
}

int withdraw(char *name,int val){
        my_stream = fopen (name, "a");
        fprintf (my_stream, "-%d\n",val);
        fclose (my_stream);
        activity += val;
}


int transfert(char *from, char *to, int val) {
        withdraw(from,val);
        deposit(to,val);
}

int balance(char *name){
        int temp=0;
        my_stream = fopen (name, "r");

        while ( fscanf(my_stream,"%s\n",buf)>0) {
                temp+=atoi(buf);
        }

        fclose (my_stream);
        return temp;
}
```

```
#include <arachne/aspect.h>

#define tax 10



aspect tax_borloo ::
        controlflow (
        int transfert(char*, char*, int),
          call (int deposit(char *name, int val)) )
            then {
                proceed("borloo", tax);
                proceed(name, val - tax);
                return 0;
                }
```

[AOSD 2005]

An aspect « fidelisation »
the sequence

```
iint deposit(char *name, int val) {
            my_stream = fopen (name, "a");
            fprintf (my_stream, "+%d\n",val);
            fclose (my_stream);
            activity += val;
}

int withdraw(char *name,int val){
            my_stream = fopen (name, "a");
            fprintf (my_stream, "-%d\n",val);
            fclose (my_stream);
            activity += val;
}


int transfert(char *from, char *to, int val) {
            withdraw(from,val);
            deposit(to,val);
}

int balance(char *name){
            int temp=0;
            my_stream = fopen (name, "r");

            while ( fscanf(my_stream,"%s\n",buf)>0) {}
                        temp+=atoi(buf);
            }

            fclose (my_stream);
            return temp;
}
```

```
#include <arachne/aspect.h>

#define tax 10


aspect fidelisation ::
          seq (
             (call (int deposit(char *name1, int val)))* )
                then {

                     }
             call (int withdraw(char *name2, int val)) )
                then {
                     if (name1 == name2)
                     proceed(nom, val-tax);
                     return 0;
                     }
```

Conclusion and future work

Related Work

- Gilk, Toskana, Dyninst …

Arachne

- First Dynamic AOSD system for C
- Without service interruption

Validation

- Squid Web Cache [IEEE Soft 2006, AOSD 2003]
- Security [AOSD 2005,PRDC 2005]
- Medical scanner Siemens (flow execution modification) [ETFA 05]
- Linux Kernel (Energy consumption) [AC 2007]

Future work

- Arachne in Hypervisor
- New abstraction for managing the program state during aspect insertion
- Operating system conception from scratch with AOSD approach
- Extension to distributed system (Grid)

Question ?