UE S1.2 - Nombres :
Comment représenter le réel ?

Nadia Abchiche-Mimouni
nadia.abchiche@u-psud.fr

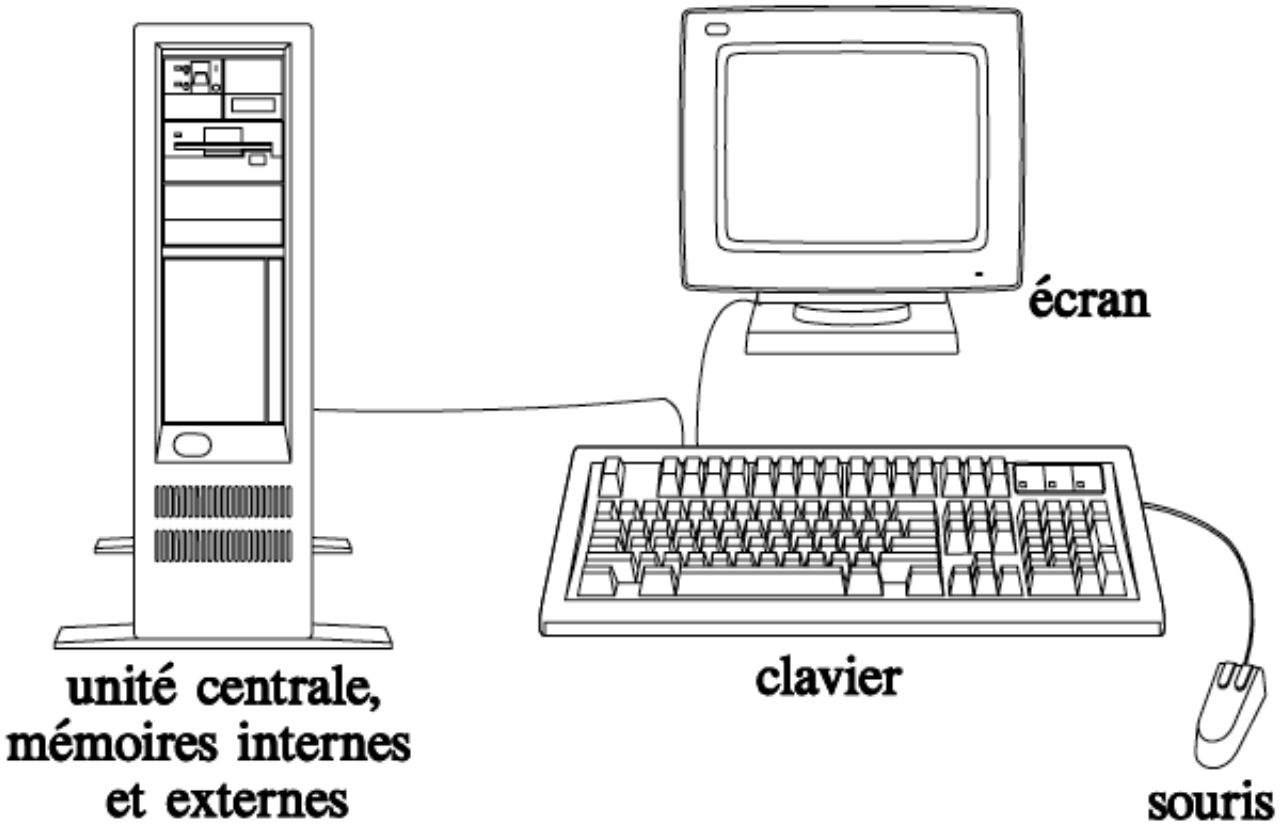
Joël Cohen
joel.cohen@villebon-charpak.fr

Séance N°1

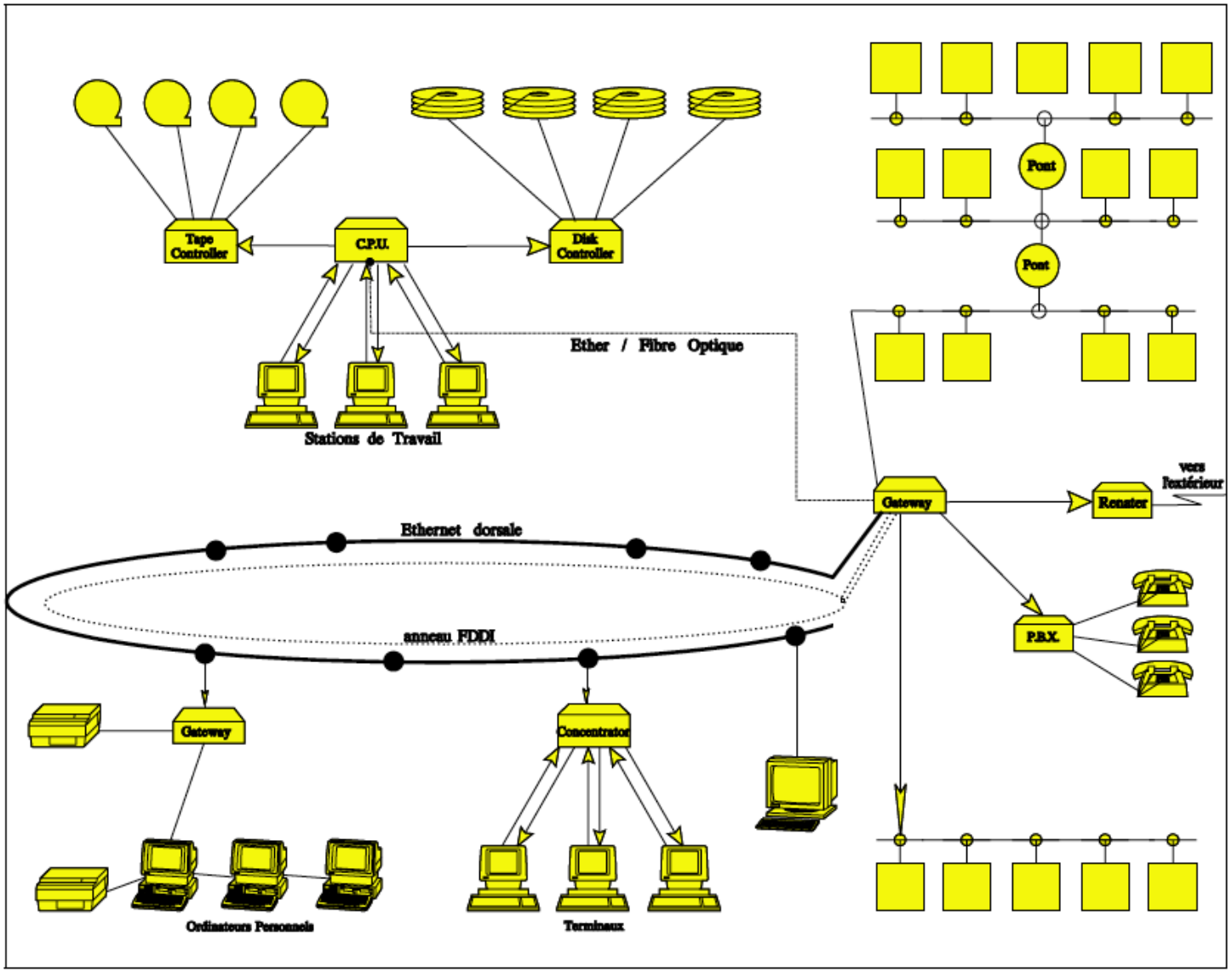
Objectifs de la séance :

- Se familiariser avec la notion « d'environnement informatique »
- Découvrir quelques commandes de base de Linux
- Notion d'algorithme et de programme

Vue externe d'une machine



Synoptique d'une installation informatique possible



Notion de couche/niveau

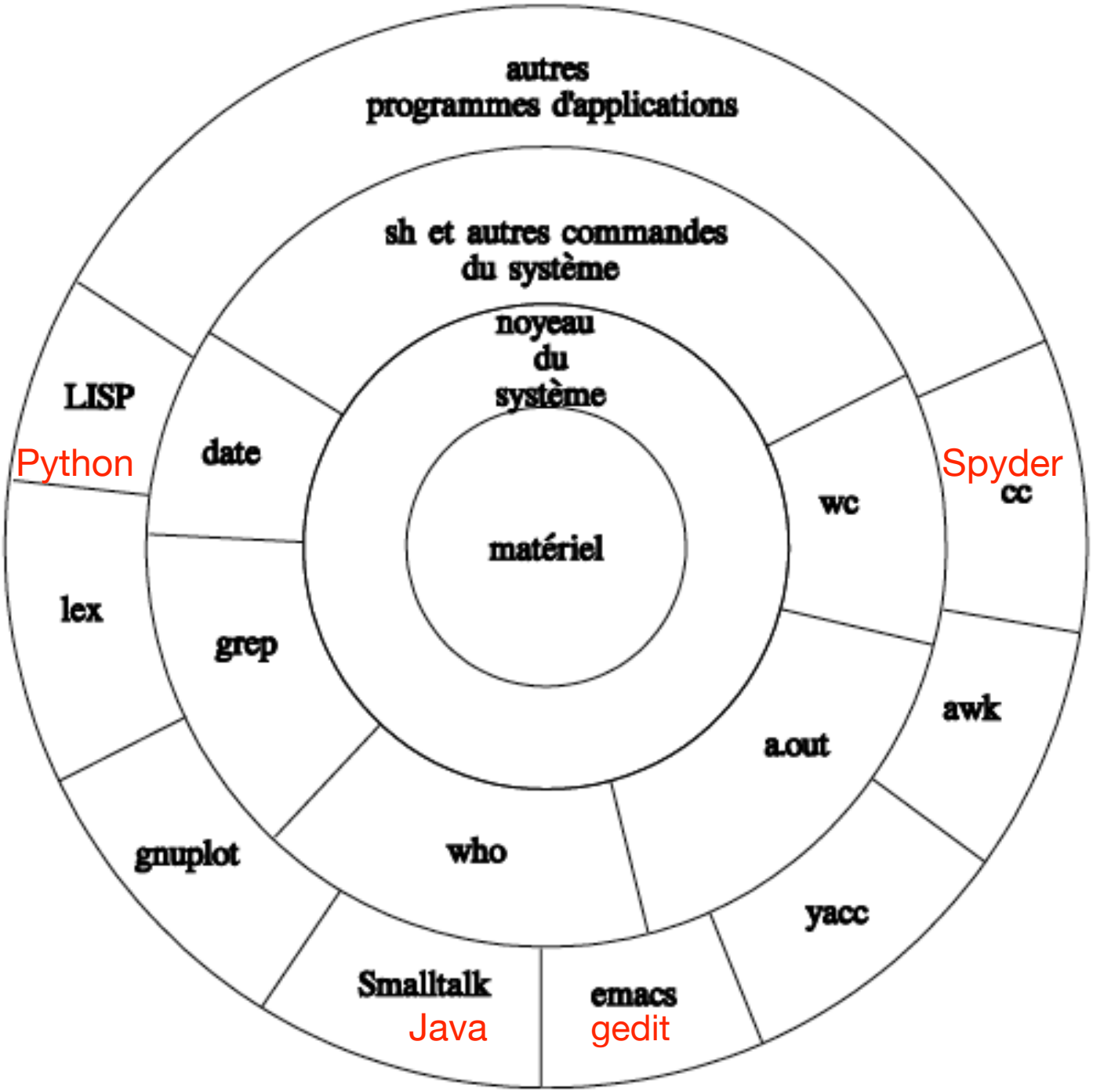
L'ensemble de programmes permettant l'utilisation d'une machine s'appelle le **système d'exploitation** ou **Operating System (OS)**

Quelques parties interagissent directement avec le matériel, où des événements peuvent se dérouler à des vitesses autour de 10^{-7} secondes (tels que le changement d'état d'une porte logique), d'autres parties ont en charge l'interaction avec les utilisateurs (où les vitesses sont plutôt de l'ordre des secondes).

Une simple frappe sur le clavier peut résulter en 10 appels à des programmes du système opératoire, en 1000 instructions machines et en 1000000 changements d'état des portes logiques.

niveau	nom	objets	exemples d'opérations
1	circuits électroniques	registres, portes, bus, etc.	clear, complémenter, transférer
2	jeu d'instructions	interprète de micro-programmes, pile d'évaluation, données	load, store, branchement, add, subtract, ...
3	procédures	segments de procédures, pile d'appels	call, return
4	interruptions, exceptions	programmes de traitement d'erreurs	invoke, mask, unmask, retry
5	processus primitifs	processus de base, sémaphores	wait, signal, suspend, resume
6	mémoire secondaire locale	canaux, blocks de données	read, write, allocate, free
7	mémoire virtuelle	segments de mémoires	read, write, fetch
8	communications	pipes, socles	create, destroy, open, close, read, write
9	système de fichiers (stockage)	fichiers	create, destroy, open, close, read, write

Notion de couche

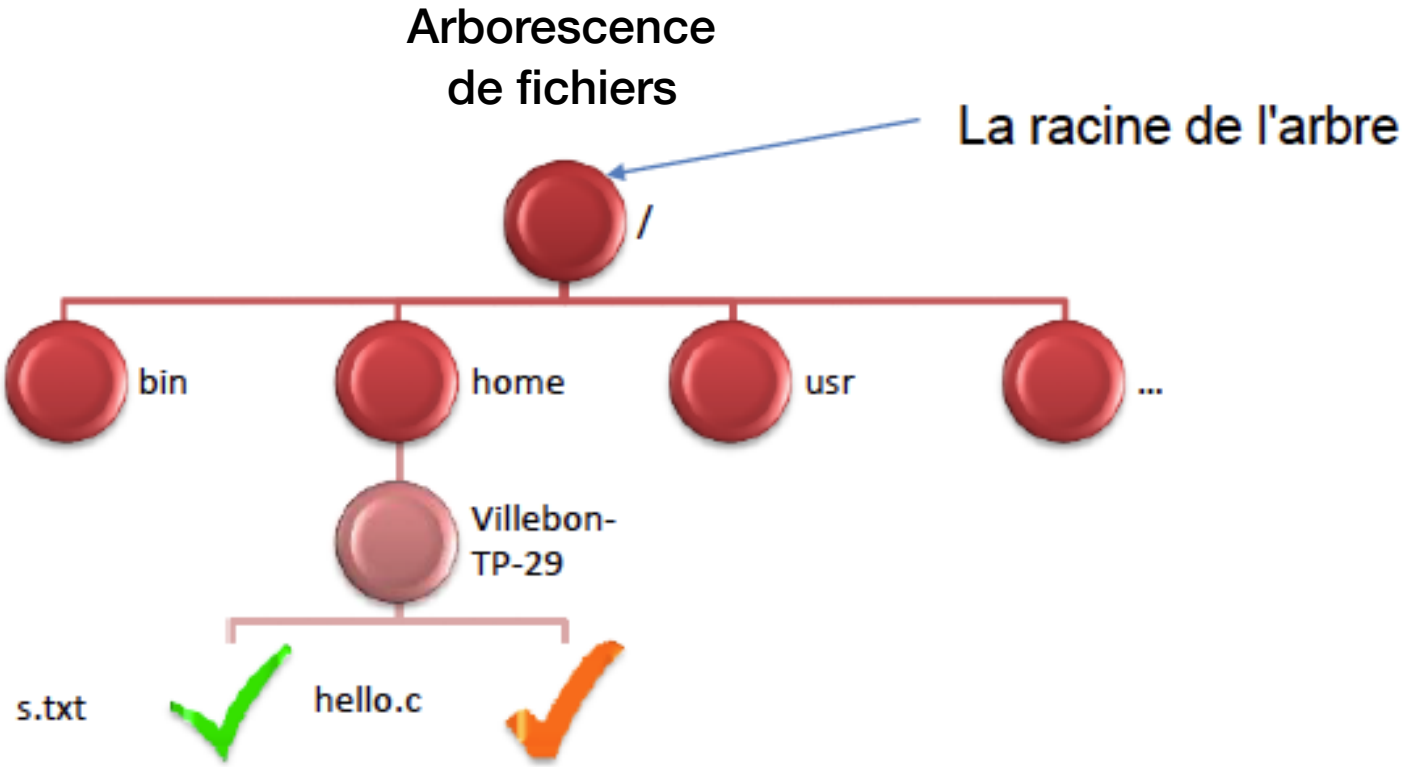


Quelques bases UNIX

- Se connecter sur une session Unix et suivre les instructions de l'enseignant-e
- Ouvrir un terminal
- Taper une commande

Arborescence de fichiers

- Les répertoires (ou dossiers – directory en anglais) et les fichiers sont organisés en arbres



Arborescence

- Vous démarrez dans votre dossier "maison" (home). Ex : Villebon-TP-29
- Commandes de navigation
  - `pwd` : Affiche le « chemin » dans l'arborescence vers le répertoire courant.
    - `/home/Villebon-TP-29`
    - Le caractère « `/` »
      - `Seul : désigne la racine de l'arbre (root)`
      - `Sinon, séparateur entre les différents niveaux du chemin`
  - `ls` : afficher la liste des fichiers du répertoire courant
  - `cd Images`
    - Se déplacer dans le dossier Images
      - Attention : la casse (majuscules / minuscules) est importante images ≠ Images ≠ iMaGes

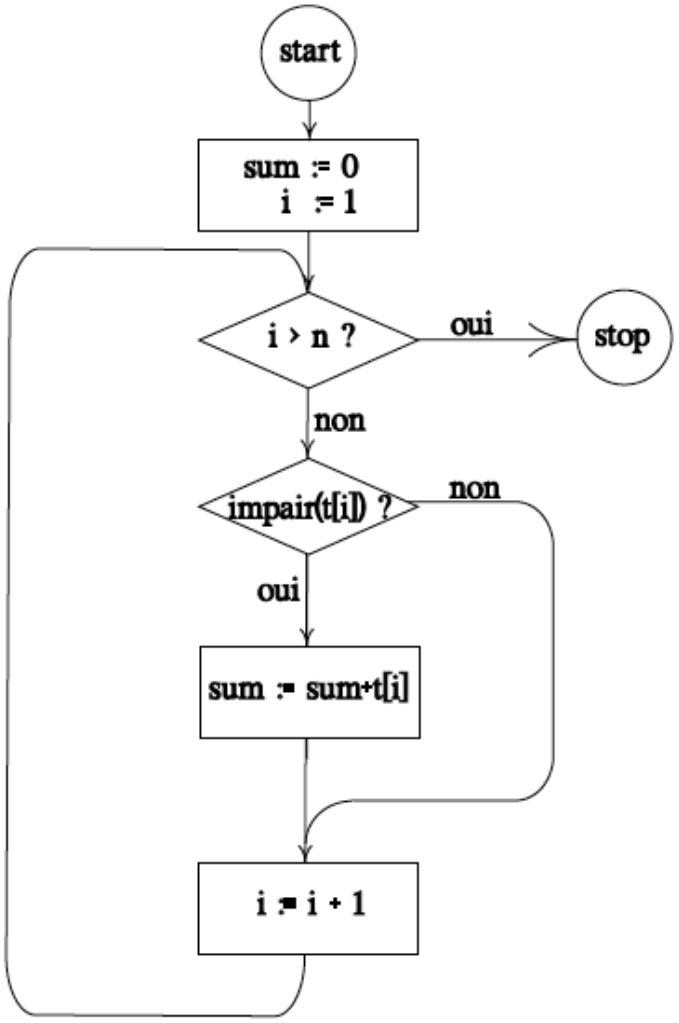
Ligne de commande : manipulations sur les fichiers

- Créer un fichier
  - `gedit s.txt`
- Afficher le contenu d'un fichier texte
  - `more d.txt`
- Copier un fichier
  - `cp s.txt d.txt`
- Créer un sous-répertoire (qui s'appelle "destination"
  - `mkdir destination`
- Déplacer / renommer un fichier
  - `renommer : mv s.txt e.txt`
  - `Déplacer : mv s.txt destination`
- Supprimer un fichier (attention, pas de corbeille => on ne peut pas annuler)
  - `rm d.txt`

Quelques entrainements

Algorithme et programme

On souhaite écrire un programme qui permet de calculer la somme des nombres impairs à partir d'une liste de nombres donnée. On suppose que ces nombres sont rangés dans des cases nommées de la façon suivante : t[Numéro de la case]



Algorithmes & programmes

Que se passerait-il si on demande au programme de faire le calcul pour 4 nombres (23, 4, 7, 34) stockés dans des cases numérotées de 1 à 4 comme ci-dessous ?

t[1] ← 23 t[1] ← 34 t[1] ← 7 t[1] ← 4

Le programme en langage machine (binaire) Motorola 68000

code machine	commentaires
00100100 01011111	adresse de retour dans A2
00100010 01011111	adresse de premier élé dans A1
00110010 00011111	compteur n dans D1
01000010 01000010	la somme dans D2 initialisé 0
01001110 11111010 00000000 00001110	saut vers fin boucle si n = 0
00001000 00101001 00000000 00000000 00000000 00000001	LOOP: si A1 est pair
01100111 00000010	alors vers SUIVANT
11010100 01010001	sinon d2 ← D2 + A1
01010100 01001001	SUIVANT: A1 prend l'élé suiv.
01010001 11001001 11111111 11110010	décrément D1, si ≠ 0 vers LOOP
00111110 10000010	D2 sur la pile
01001110 11010010	retour vers appelant

En langage assembleur Motorola 68000

Algorithmes & programmes

étiquettes	instructions		commentaires
SUMODD:	MOVE.L	(A7)+,A2	dépile adresse retour dans A2
	MOVE.L	(A7)+,A1	dépile adresse du premier élément dans A1
	MOVE.W	(A7)+,D1	dépile le nombre d'éléments n dans D1
	CLR.W	D2	mettre D2, la somme, à 0
	JMP	COUNT	saut vers la fin de boucle pour tester si n = 0
LOOP:	BTST	0,1(A1)	si l'élément adressé par A1 est pair
	BEQ.S	SUIVANT	alors continuer à SUIVANT
	ADD.W	(A1),D2	sinon ajouter l'élément à D2
SUIVANT:	ADDQ.W	#2,A1	mettre dans A1 l'adresse de l'élément suivant
COUNT:	DBF	D1, LOOP	décrémenter D1; sauf s'il est = -1 continuer en LOOP
	MOVE.W	D2,-(A7)	empiler la somme contenue dans D2
	JMP	(A2)	retourner vers l'adresse contenue dans A2

Algorithmes & programmes

Le programme en langage BASIC

```
0100 DIM T(100)
0200 READ N
0300 FOR I = 1 TO N
0400   READ T(I)
0500 NEXT I
0600 GOSUB 1100
0700 PRINT S
0800 GOTO 2000
0900 DATA 4
1000 DATA 23, 34, 7, 9
1100 REM S SERA LA SOMME DES ELEMENTS IMPAIRS DE T(I)
1200 LET S = 0
1300 FOR I - 1 TO N
1400    IF NOT ODD(T(I)) THEN GOTO 1600
1500   LET S = S + T(I)
1600 NEXT I
1700 RETURN
2000 END
```

Quelques extraits écrit en langage Cobol, (armée américaine)

Algorithmes & programmes

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 NUMERIC-VARIABLES USAGE IS COMPUTATIONAL.
  01 TERMS PICTURE 9999 OCCURS 100 TIMES INDEXED BY I.
  02 N PICTURE 999.
  02 SUM PICTURE 999999.
  02 HALF-TERM PICTURE 9999.
  02 RMDR PICTURE 9.

PROCEDURE DIVISION.
EXAMPLE.
  MOVE 23 TO TERMS(1).
  MOVE 34 TO TERMS(2).
  MOVE 7 TO TERMS(3).
   MOVE 9 TO TERMS(4).
   MOVE 4 TO N.
 PERFORM SUM-ODDS.

SUM-ODDS.
 MOVE 0 TO SUM.
 PERFORM TAKE-ONE-TERM VARYING I FROM 1 BY 1 UNTIL I > N.
TAKE-ONE-TERM.
 DIVIDE 2 INTO TERMS(I) GIVING HALF-TERM REMAINDER RMDR.
 IF RMDR IS EQUAL TO 1: ADD TERMS(I) TO SUM.
```

Le programme écrit en langage Pascal

Algorithmes & programmes

```
program SumOddNumbers;
type Termindex = 1..100;
TermArray = array[Termindex] of integer;
var myTerms: TermArray;
function SumOdds (n: Termindex; terms: TermArray): integer;
var i: Termindex;
sum: integer;
begin
sum := 0;
for i:= 1 to n do
if Odd(terms[i]) then
sum := sum + terms[i];
SumOdds = sum;
end;
begin
myTerms[1]:23; myTerms[2] := 34; myTerms[3] := 7; myTerms[4] := 9;
WriteLn (SumOdds (4, myTerms));
end.
```

Nicolas Wirth, fin des années 1960

Le programme écrit en langage FORTH

Algorithmes & programmes

```
:SUMODDS
0 SWAP 0
DO
SWAP DUP 2 MOD
IF +
ELSE DROP
THEN
LOOP
:
```

Charles H. Moore en 1970, pour les calculs scientifiques (astronomie)

Le programme écrit en langage LISP

Algorithmes & programmes

```
(DE SUMODDS (TERMS)
(COND
((NULL TERMS) 0)
((ODD? (CAR TERMS)) (+ (CAR TERMS)(SUMODDS (CDR TERMS))))
(T (SUMODDS (CDR TERMS)))))
```

Par John McCarthy en 1959 au MIT pour l'Intelligence Artificielle

Le programme écrit en langage Smalltalk

Algorithmes & programmes

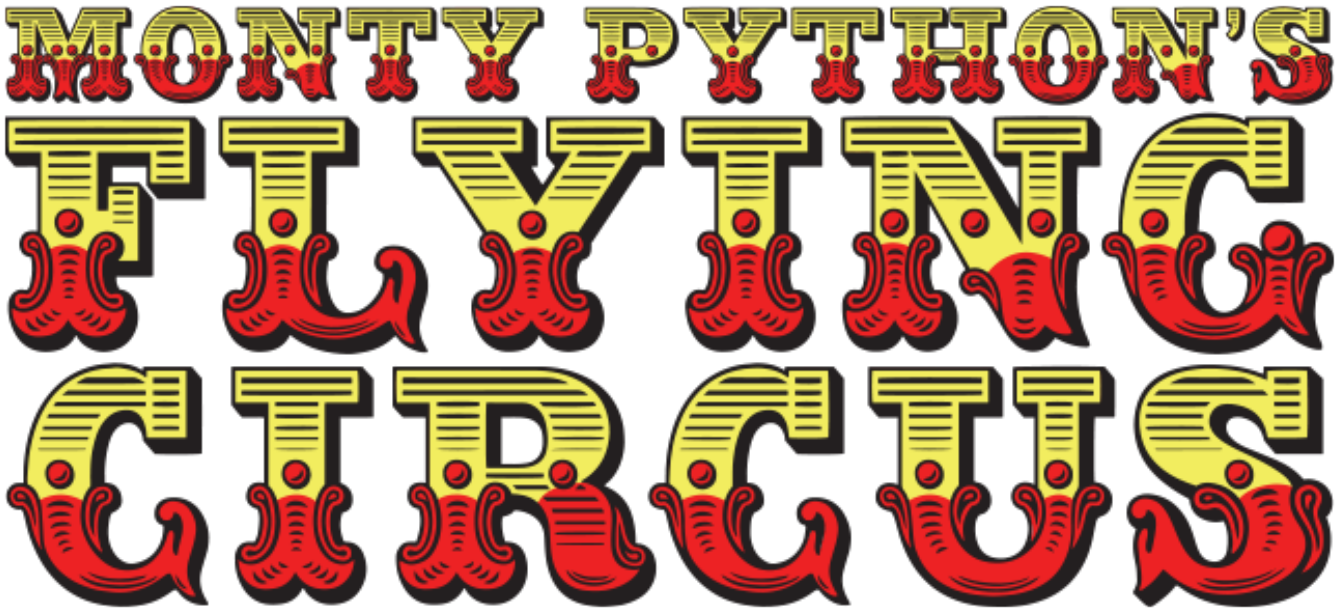
```
sumOfOddNumbers
“d’abord un exemple d’appel :”
“#(23 34 7 9) sumOfOddNumbers”
^ self inject: 0 into: [:x :y | x + (y odd
ifTrue: [y]
ifFalse: [0])]
```

Par Alan Kay en 1970-72 au Xerox Palo Alto Research Center

ADA, C, PROLOG, JAVA, PERL, ...

Algorithmes & programmes

De nombreux autres langages encore …. et bien sûr **PYTHON !**



Bibliographie/webographie

Ce support de cours a été réalisé avec l'aide des supports de cours de collègues et auteurs :

Harald Wertz :

- Algorithmes et programmes : http://www.ai.univ-paris8.fr/~hw/unx4.pdf
- Machines et Systèmes : http://www.ai.univ-paris8.fr/~hw/unx1.pdf
- Système de fichier : http://www.ai.univ-paris8.fr/~hw/unx2.pdf

Isabelle Demeure, Valerie Gautard : voir sur Dokeos

Ressources et documentation

- http://firecontrolman.tpub.com/14100/css/Magnetic-Tape-Controller-242.htm, tape controller