

A decorative network diagram in the top-left corner, consisting of a complex web of interconnected nodes and lines, rendered in a light gray color. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting a hierarchical or central structure. The lines are thin and connect the nodes in a non-linear fashion.

# OS Lab 2: Adding User Programs to XV6

Farah Kabesh - 900191706

Karim Sherif - 900191474

A decorative network diagram in the bottom-right corner, similar to the one in the top-left, featuring a complex web of interconnected nodes and lines in a light gray color. The nodes are small circles, some with concentric circles, connected by thin lines in a non-linear fashion.

# Role of Team Members

## Farah

- © Selection Sort Program
- © Changes in Makefile
- © Presentation


## Karim

- © Linear Search Program
- © Bubble Sort Program
- © Changes in Makefile
- © Changes in user.h and ulib.c files

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are highlighted with a double-circle outline. The lines are thin and gray, creating a mesh-like structure.

# 3 User Programs

Linear Search  
Selection Sort  
Bubble Sort

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It shows a cluster of nodes connected by lines, with some nodes having a double-circle outline. The overall style is minimalist and technical.

# Linear Search

- © Finds whether a desired element  $x$  is present in an array.
- © Uses a sequential search algorithm.
- © Starts from the first element in the array and compares the element we're searching for ( $x$ ) with each element in the array until it is found or the array ends.
  - If  $x$  is found, it returns its index.
  - Otherwise, it will return -1.

# Pseudo-code for Linear Search

```
LINEAR_SEARCH (array, size, value)
  for each element in the array
    if element == value
      return the element's index
  else
    return -1
```

# Testing the Program

```
SeaBIOS (version 1.13.0-1ubuntu1.1)
```

```
iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CB00+1FECCB00 CA00
```

```
Booting from Hard Disk...
```

```
cpu1: starting 1
```

```
cpu0: starting 0
```

```
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
```

```
init: starting sh
```

```
$ linearS 45 87 62 31 98 21 62
```

```
Item is present at index 3
```

```
$ _
```

# Selection Sort

- © Sorting algorithm which finds the smallest element in an array of unsorted elements and places it at the beginning of the array.
- © Proceeds to do this repeatedly by iterating through the rest of the unsorted elements in the array till they are correctly placed and sorted in ascending order.
- © Relies on the concept of swapping elements.

# Pseudo-code for Selection Sort

```
selectionSort (array, size)
  for (size-1) times
    set first element as minimum
    for each element in unsorted array
      if element < minimum
        set element as new minimum
    swap new minimum with first element
```



# Testing the Program

```
SeaBIOS (version 1.13.0-1ubuntu1.1)
```

```
iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CB00+1FECCB00 CA00
```

```
Booting from Hard Disk...
```

```
cpu1: starting 1
```

```
cpu0: starting 0
```

```
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
```

```
init: starting sh
```

```
$ selectionsort 45 21 -5 17 -2 87 61
```

```
Sorted array in Ascending Order using SelectionSort:
```

```
-5 -2 17 21 45 61 87
```

```
$
```

# Bubble Sort

- ⦿ Sorting algorithm which starts with the first 2 elements of an array.
- ⦿ The elements are swapped if the first element is greater than the second.
- ⦿ The second and third elements are then compared and are swapped if they're in the wrong order.
- ⦿ Iterates through all the elements by repeating the above process till the last unsorted element in the array.
- ⦿ Array is sorted when all elements are in the correct positions in ascending order.
- ⦿ Relies on the concept of swapping elements.

# Pseudo-code for Bubble Sort

```
bubbleSort (array, size)
  for step  $\leftarrow$  0 to size -1
    for i  $\leftarrow$  1 to size - step -1
      if array[i] > array[i+1]
        temp = array[i]
        array[i] = array[i+1]
        array[i+1] = temp
```

# Testing the Program

```
SeaBIOS (version 1.13.0-1ubuntu1.1)
```

```
iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CB00+1FECCB00 CA00
```

```
Booting from Hard Disk...
```

```
cpu1: starting 1
```

```
cpu0: starting 0
```

```
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
```

```
init: starting sh
```

```
$ bubblesort 45 87 50 25 71 17
```

```
Sorted Array in Ascending Order using BubbleSort:
```

```
17 25 45 50 71 87
```

```
$
```

# Changes & Additions Made in XV6

- © The 3 user programs for the sort and search algorithms were created, written in the C programming language and saved inside the XV6 OS folder as:

- `linearS.c`
- `selectionsort.c`
- `bubblesort.c`

-----

- © The files containing the C programs had to contain the following header files:

- `include "types.h"`
- `include "stat.h"`
- `include "user.h"`

# Changes & Additions Made in XV6

- © The makefile in the XV6 folder was edited in order to add them to the source code of XV6 and be able to compile the programs.
- © The names of the files containing the C programs were added to the following 2 sections:

```
168 UPROGS=\
169     _cat\
170     _echo\
171     _forktest\
172     _grep\
173     _init\
174     _kill\
175     _ln\
176     _ls\
177     _mkdir\
178     _rm\
179     _sh\
180     _stressfs\
181     _usertests\
182     _wc\
183     _zombie\
184     _linearS\
185     _selectionsort\
186     _bubblesort\
187
```

```
254 EXTRA=\
255     mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
256     ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c \
257     linearS.c selectionsort.c bubblesort.c\
258     README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
259     .gdbinit.tmpl gdbutil\
260
```

# Changes & Additions Made in XV6

- © Created a function `int atoi2(char *s)` and added it to the `user.h` and `ulib.c` files in order to handle negative numbers in the user programs.

```
Open  ▾  ~ / xv6  *user.h
1 struct stat;
2 struct rtcdate;
3
4 // system calls
5 int fork(void);
6 int exit(void) __attribute__((noreturn));
7 int wait(void);
8 int pipe(int*);
9 int write(int, const void*, int);
10 int read(int, void*, int);
11 int close(int);
12 int kill(int);
13 int exec(char*, char**);
14 int open(const char*, int);
15 int mknod(const char*, short, short);
16 int unlink(const char*);
17 int fstat(int fd, struct stat*);
18 int link(const char*, const char*);
19 int mkdir(const char*);
20 int chdir(const char*);
21 int dup(int);
22 int getpid(void);
23 char* sbrk(int);
24 int sleep(int);
25 int uptime(void);
26
27 // ulib.c
28 int stat(const char*, struct stat*);
29 char* strcpy(char*, const char*);
30 void* memmove(void*, const void*, int);
31 char* strchr(const char*, char c);
32 int strcmp(const char*, const char*);
33 void printf(int, const char*, ...);
34 char* gets(char*, int max);
35 uint strlen(const char*);
36 void* memset(void*, int, uint);
37 void* malloc(uint);
38 void free(void*);
39 int atoi(const char*);
40 int atoi2(char *s);
```

```
Open  ▾  ~ / xv6  ulib.c
75
76 fd = open(n, O_RDONLY);
77 if(fd < 0)
78     return -1;
79 r = fstat(fd, st);
80 close(fd);
81 return r;
82 }
83
84 int
85 atoi(const char *s)
86 {
87     int n;
88
89     n = 0;
90     while('0' <= *s && *s <= '9')
91         n = n*10 + *s++ - '0';
92     return n;
93 }
94
95 void*
96 memmove(void *vdst, const void *vsrc, int n)
97 {
98     char *dst;
99     const char *src;
100
101     dst = vdst;
102     src = vsrc;
103     while(n-- > 0)
104         *dst++ = *src++;
105     return vdst;
106 }
107
108 int
109 atoi2(char *s)
110 {
111     int n;
112
113     n = 0;
114     while('0' <= *s && *s <= '9')
115         n = n*10 + *s++ - '0';
116     return n;
117 }
```