

Advanced Unix

Assignment 1

1. In the first part, in order to pass information via files, I created input files with 7 random numbers. After asking for the number of processes to be created, the child prints its cid number. Then, I am printing the numbers that are read from the input file. Child processes open only related input files such as first child opens and reads input1.txt and second child opens and reads input2.txt etc. Other than creating related output[1, 2, ..].txt files, I also created myoutput[1,2, ..].txt files which are same as the output[1, 2, ..].txt files but the order of the information is changed. It is done to easily read the information needed to be written to the output.txt file in the correct order.

In order to sort with respect to the execution time of the child processes, I created a separate array called as "execTimeArray". After sorting the contents of the array, I compare the elements of the array with the execution time information of the myoutput[1, 2, ..].txt file. Inside myoutput[1, 2, ..].txt file, the execution time information can be found in the first line so it becomes easy to parse the file to access to the required information. After finding the required execution time, then I am writing the contents of the myoutput[1, 2, ..].txt file to the output.txt file in the same order. "fnum" shows the execution time information which is a float number that is retrieved from the related myoutput[1, 2, ..].txt file.

In my first example, I have created one child. I also implemented the signal support which can be seen in the screenshots. The screenshots from the command line, the input1.txt, output1.txt, myoutput1.txt and output.txt files can be seen below:

```
Enter the number of child processes to create: 1
Hello I am child with id: 4644
Contents of the file input1.txt:
Array Elements:
47 2 12 42 9 95 23
Array Elements After Sorting:
2 9 12 23 42 47 95
Execution time:0.000005
killing cid:4644
execTimeArray: 2.000005 fnum:2.000005
exectimearray[j]: 2.000005 and fnum: 2.000005
signal name:SIGUSR2
signal receive time:17:22:13
```

```
input1.txt x
47 2 12 42 9 95 23|
```

```
output1.txt x my
2 9 12 23 42 47 95
2.000005
SIGUSR2 17:22:13
```

```
myoutput1.txt x ou
2.000005
2 9 12 23 42 47 95
SIGUSR2
17:22:13|
```

```
output.txt x myoutput1.txt x output1.txt x
2.000005 2 9 12 23 42 47 95 SIGUSR2 17:22:13
```

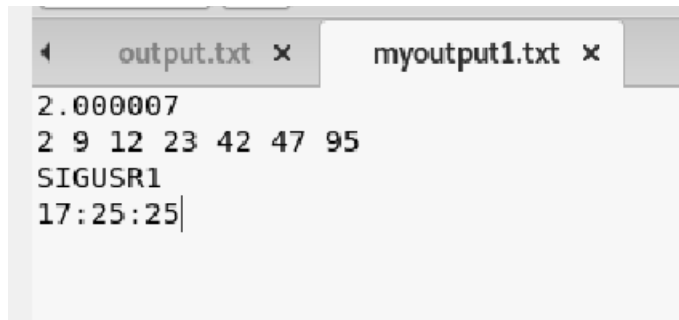
In my second example, I have created two child processes. The screenshots from the command line, the input1.txt, the input2.txt, output1.txt, output2.txt, myoutput1.txt, myoutput2.txt and output.txt files can be seen below:

```
Enter the number of child processes to create: 2
Hello I am child with id: 4661
  Contents of the file input1.txt:
Array Elements:
47 2 12 42 9 95 23
Array Elements After Sorting:
2 9 12 23 42 47 95
Execution time:0.000007
  killing cid:4661
Hello I am child with id: 4665
  Contents of the file input2.txt:
Array Elements:
4 56 1 62 33 8 26
Array Elements After Sorting:
1 4 8 26 33 56 62
Execution time:0.000003
  killing cid:4665
execTimeArray: 2.000003 2.000007 fnum:2.000007
execTimearray[j]: 2.000003 and fnum: 2.000007
fnum:2.000003
execTimearray[j]: 2.000003 and fnum: 2.000003
signal name:SIGUSR1
signal receive time:17:25:25
fnum:2.000007
execTimearray[j]: 2.000007 and fnum: 2.000007
signal name:SIGUSR1

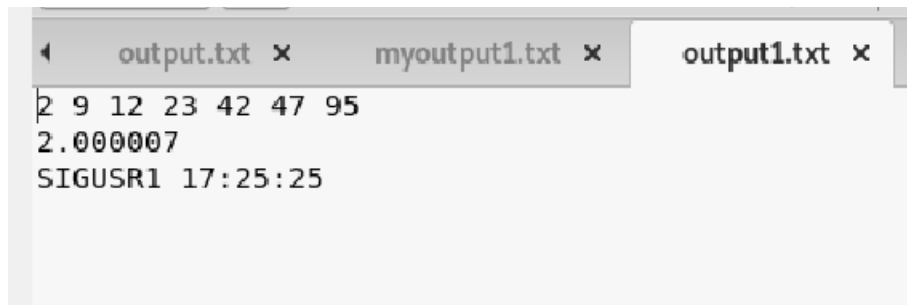
execTimearray[j]: 2.000007 and fnum: 2.000007
signal name:SIGUSR1
signal receive time:17:25:25
fnum:2.000003
execTimearray[j]: 2.000007 and fnum: 2.000003
exit(0);
```

```
input2.txt x outp
4 56 1 62 33 8 26
```

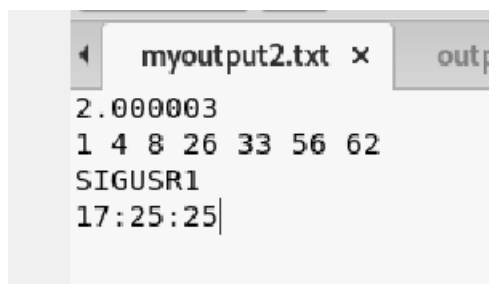
```
output.txt x myoutput1.txt x output1.txt x
2.000003 1 4 8 26 33 56 62 SIGUSR1 17:25:25
2.000007 2 9 12 23 42 47 95 SIGUSR1 17:25:25
```



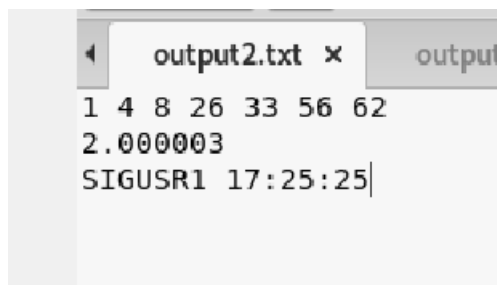
```
output.txt x myoutput1.txt x
2.000007
2 9 12 23 42 47 95
SIGUSR1
17:25:25|
```



```
output.txt x myoutput1.txt x output1.txt x
2 9 12 23 42 47 95
2.000007
SIGUSR1 17:25:25
```



```
myoutput2.txt x output
2.000003
1 4 8 26 33 56 62
SIGUSR1
17:25:25|
```



```
output2.txt x output
1 4 8 26 33 56 62
2.000003
SIGUSR1 17:25:25|
```

In my third example, I have created three child processes. The screenshots from the command line and output.txt files can be seen below. The other txt files are not included because they are similar to the examples above:

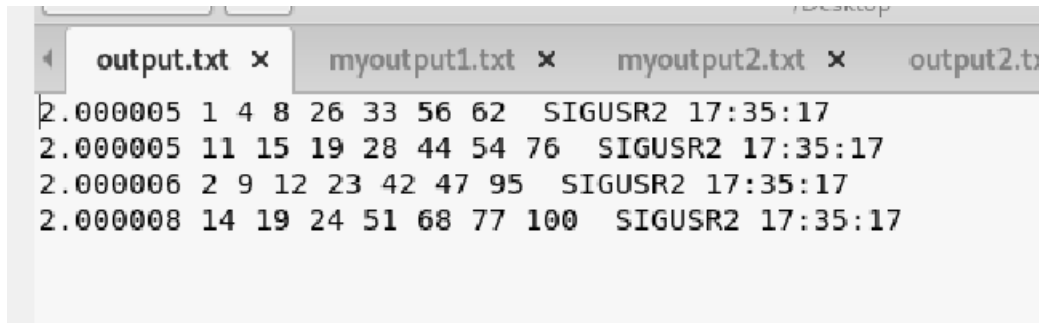
```
Enter the number of child processes to create: 3
Hello I am child with id: 4691
  Contents of the file input1.txt:
Array Elements:
47 2 12 42 9 95 23
Array Elements After Sorting:
2 9 12 23 42 47 95
Execution time:0.000005
  killing cid:4691
Hello I am child with id: 4695
  Contents of the file input2.txt:
Array Elements:
4 56 1 62 33 8 26
Array Elements After Sorting:
1 4 8 26 33 56 62
Execution time:0.000006
  killing cid:4695
Hello I am child with id: 4701
  Contents of the file input3.txt:
Array Elements:
100 24 19 14 77 51 68
Array Elements After Sorting:
14 19 24 51 68 77 100
Execution time:0.000010
  killing cid:4701
execTimeArray: 2.000005 2.000006 2.000010 fnum:2.000005
exectimearray[j]: 2.000005 and fnum: 2.000005
```

```
execTimeArray: 2.000005 2.000006 2.000010 fnum:2.000005  
exectimearray[j]: 2.000005 and fnum: 2.000005  
signal name:SIGUSR1  
signal receive time:17:31:48  
fnum:2.000006  
exectimearray[j]: 2.000005 and fnum: 2.000006  
fnum:2.000010  
exectimearray[j]: 2.000005 and fnum: 2.000010  
fnum:2.000005  
exectimearray[j]: 2.000006 and fnum: 2.000005  
fnum:2.000006  
exectimearray[j]: 2.000006 and fnum: 2.000006  
signal name:SIGUSR1  
signal receive time:17:31:48  
fnum:2.000010  
exectimearray[j]: 2.000006 and fnum: 2.000010  
fnum:2.000005  
exectimearray[j]: 2.000010 and fnum: 2.000005  
fnum:2.000006  
exectimearray[j]: 2.000010 and fnum: 2.000006  
fnum:2.000010  
exectimearray[j]: 2.000010 and fnum: 2.000010  
signal name:SIGUSR1  
signal receive time:17:31:48
```

```
exectimearray[j]: 2.000010 and fnum: 2.000010  
signal name:SIGUSR1  
signal receive time:17:31:48
```

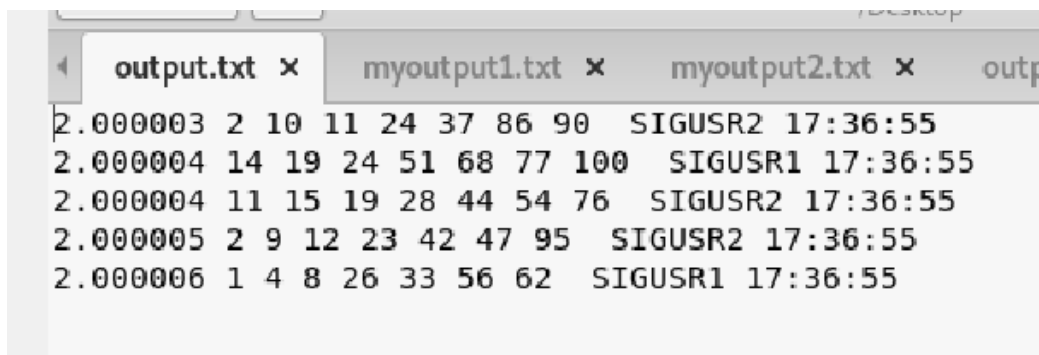
```
output.txt x myoutput1.txt x myoutput2.txt x output  
2.000005 2 9 12 23 42 47 95 SIGUSR1 17:31:48  
2.000006 1 4 8 26 33 56 62 SIGUSR1 17:31:48  
2.000010 14 19 24 51 68 77 100 SIGUSR1 17:31:48
```

In my fourth example, I have created four child processes. I also check if the “execTimeArray” has duplicate entries in order not to write twice to the output.txt file. If the “execTimeArray” array has duplicates, I remove it from the array. The example below includes two processes that have the same execution time. The screenshots from the command line and output.txt files can be seen below. The other txt files are not included because they are similar to the examples above:



```
output.txt x myoutput1.txt x myoutput2.txt x output2.txt
2.000005 1 4 8 26 33 56 62 SIGUSR2 17:35:17
2.000005 11 15 19 28 44 54 76 SIGUSR2 17:35:17
2.000006 2 9 12 23 42 47 95 SIGUSR2 17:35:17
2.000008 14 19 24 51 68 77 100 SIGUSR2 17:35:17
```

In my fifth example, I have created five child processes. The screenshot from the output.txt files can be seen below. The other txt files and the command line are not included because they are similar to the examples above:



```
output.txt x myoutput1.txt x myoutput2.txt x outp
2.000003 2 10 11 24 37 86 90 SIGUSR2 17:36:55
2.000004 14 19 24 51 68 77 100 SIGUSR1 17:36:55
2.000004 11 15 19 28 44 54 76 SIGUSR2 17:36:55
2.000005 2 9 12 23 42 47 95 SIGUSR2 17:36:55
2.000006 1 4 8 26 33 56 62 SIGUSR1 17:36:55
```

2. In the second part, in order to pass information via pipes, I created random arrays containing 7 elements after getting the number of child processes to be created from the user. Then, I am printing the unsorted and sorted random arrays. Later, after passing the arrays through the related pipes, child processes start to work on the arrays and send the required information through the related pipe. Parent prints the information received from the child processes to the screen and also to the outputpipe.txt file. In here, I also have a separate "execTimeArray" array in order to sort the child processes information with respect to their execution time.

In my first example, I have created one child. I also implemented the signal support which can be seen in the screenshots. The screenshots from the command line, the input1.txt, output1.txt and outputpipe.txt files can be seen below:


```
Enter the number of child processes to create: 1
Hello I am child with id: 4870
  Array Elements Before Sorting:
84 87 78 16 94 36 87
Array Elements After Sorting:
16 36 78 84 87 87 94

Execution time:0.000005
Signal receive time:17:55:27
parent reads execTime: 2.000005
parent reads signal name:SIGUSR2
parent reads signal receive time:17:55:27
Printing exec time array:
2.000005
Results:
2.000005 16 36 78 84 87 87 94 SIGUSR2 17:55:27
```

outputpipe.txt	x	output.txt	x
2.000005	16 36 78 84 87 87 94	SIGUSR2	17:55:27

In my second example, I have created two children processes. The screenshots from the command line, the input1.txt and outputpipe.txt files can be seen below:

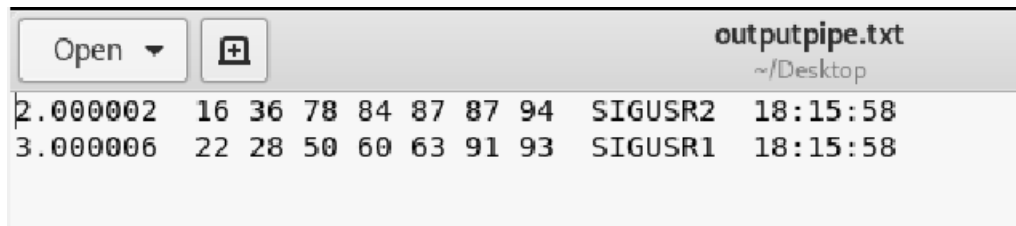
```
Enter the number of child processes to create: 2
Hello I am child with id: 1844
  Array Elements Before Sorting:
84 87 78 16 94 36 87
Array Elements After Sorting:
16 36 78 84 87 87 94

Execution time:0.000002
Signal receive time:18:15:58
parent reads execTime: 2.000002
parent reads signal name:SIGUSR2
parent reads signal receive time:18:15:58
Hello I am child with id: 1845
  Array Elements Before Sorting:
93 50 22 63 28 91 60
Array Elements After Sorting:
22 28 50 60 63 91 93

Execution time:0.000006
Signal receive time:18:15:58
parent reads execTime: 3.000006
parent reads signal name:SIGUSR1
```



```
parent reads signal name:SIGUSR1
parent reads signal receive time:18:15:58
Printing exec time array:
2.000002 3.000006
Results:
2.000002 16 36 78 84 87 87 94 SIGUSR2 18:15:58
3.000006 22 28 50 60 63 91 93 SIGUSR1 18:15:58
```



```
2.000002 16 36 78 84 87 87 94 SIGUSR2 18:15:58
3.000006 22 28 50 60 63 91 93 SIGUSR1 18:15:58
```

In my third example, I have created three children processes. The screenshots from the command line, the input1.txt and outputpipe.txt files can be seen below:

```
Enter the number of child processes to create: 3
Hello I am child with id: 2298
Array Elements Before Sorting:
84 87 78 16 94 36 87
Array Elements After Sorting:
16 36 78 84 87 87 94

Execution time:0.000004
Signal receive time:19:3:49
parent reads execTime: 2.000004
parent reads signal name:SIGUSR2
parent reads signal receive time:19:3:49
Hello I am child with id: 2299
Array Elements Before Sorting:
93 50 22 63 28 91 60
Array Elements After Sorting:
22 28 50 60 63 91 93

Execution time:0.000006
Signal receive time:19:3:49
```

```
Execution time:0.000006
Signal receive time:19:3:49
parent reads execTime: 3.000006
parent reads signal name:SIGUSR1
parent reads signal receive time:19:3:49
Hello I am child with id: 2300
Array Elements Before Sorting:
64 27 41 27 73 37 12
Array Elements After Sorting:
12 27 27 37 41 64 73

Execution time:0.000005
Signal receive time:19:3:49
parent reads execTime: 7.000005
parent reads signal name:SIGUSR2
parent reads signal receive time:19:3:49
Printing exec time array:
2.000004 3.000006 7.000005
Printing exec time array after sort:
2.000004 3.000006 7.000005
Results:
2.000004 16 36 78 84 87 87 94 SIGUSR2 19:3:49
3.000006 22 28 50 60 63 91 93 SIGUSR1 19:3:49
7.000005 12 27 27 37 41 64 73 SIGUSR2 19:3:49
```

outputpipe.txt										x
2.000004	16	36	78	84	87	87	94	SIGUSR2	19:3:49	
3.000006	22	28	50	60	63	91	93	SIGUSR1	19:3:49	
7.000005	12	27	27	37	41	64	73	SIGUSR2	19:3:49	

In my fourth example, I have created four children processes. The screenshots from the command line, the input1.txt and outputpipe.txt files can be seen below. The command line output is not included because it is similar to the examples above:

```
Results:
1.000004 24 30 31 63 68 69 83 SIGUSR1 19:11:21
2.000006 16 36 78 84 87 87 94 SIGUSR2 19:11:21
3.000004 22 28 50 60 63 91 93 SIGUSR1 19:11:21
7.000005 12 27 27 37 41 64 73 SIGUSR2 19:11:21
root@kali: ~/Desktop#
```

outputpipe.txt										×
1.0000004	24	30	31	63	68	69	83	SIGUSR1	19:11:21	
2.0000006	16	36	78	84	87	87	94	SIGUSR2	19:11:21	
3.0000004	22	28	50	60	63	91	93	SIGUSR1	19:11:21	
7.0000005	12	27	27	37	41	64	73	SIGUSR2	19:11:21	

In my fifth example, I have created five children. The screenshots from the command line, the input1.txt and outputpipe.txt files can be seen below. The command line output is not included because it is similar to the examples above:

```
Results:
1.0000005 24 30 31 63 68 69 83 SIGUSR1 19:12:48
2.0000004 3 23 30 36 59 68 70 SIGUSR2 19:12:48
2.0000005 16 36 78 84 87 87 94 SIGUSR2 19:12:48
3.0000004 22 28 50 60 63 91 93 SIGUSR1 19:12:48
7.0000005 12 27 27 37 41 64 73 SIGUSR2 19:12:48
root@kali:~/Desktop#
```

outputpipe.txt										×
1.0000005	24	30	31	63	68	69	83	SIGUSR1	19:12:48	
2.0000004	3	23	30	36	59	68	70	SIGUSR2	19:12:48	
2.0000005	16	36	78	84	87	87	94	SIGUSR2	19:12:48	
3.0000004	22	28	50	60	63	91	93	SIGUSR1	19:12:48	
7.0000005	12	27	27	37	41	64	73	SIGUSR2	19:12:48	