

3. Modify to take several command line arguments through getopt. Program should take three options, either -n x, -h, or -p.

The -h option should display a help message indicating the type of input it expects and then the program should terminate

If it takes -n x, it should store the x into a variable for use for the rest of the program. If it receives -p as an option it should use perror to generate a test error message as described in next task.

```
[festerva@hoare7 festerva.1]$ gcc problem3.c
[festerva@hoare7 festerva.1]$ ./a.out -p
-1299642395 error:Success
[festerva@hoare7 festerva.1]$ vim problem3.c
[festerva@hoare7 festerva.1]$ gcc problem3.c
[festerva@hoare7 festerva.1]$ ./a.out -p
./a.out error:Success
[festerva@hoare7 festerva.1]$ ./a.out -h
Please enter n and an integer value (For example n 10)
.1]$ ./a.out -n 5
i:1 process ID:2606 parent ID:7164 child ID:2607
i:2 process ID:2607 parent ID:2606 child ID:2608
i:3 process ID:2608 parent ID:2607 child ID:2609
i:4 process ID:2609 parent ID:2608 child ID:2610
i:5 process ID:2610 parent ID:2609 child ID:0
Aborted
[festerva@hoare7 festerva.1]$
```

4. Generate error messages by my_prog: Error: Detailed error message using perror. Where my_prog is the name of the executable

(argv[0] that you are trying to execute. This should not be hardcoded. */

5.

1

/* Run program 3.1 and observe results for different numbers of processes.

The i numbers each process. ID seems to start with 17 and have different numbers depending on the process. But, the next one seems to be in order. The parent ID seems to start with the same address. Child ID starts with 17. Every once in a while there is a festerva@hoare line. Seems to be a parent ID of 1 on occasion.

```
[festerva@hoare7 festerva.1]$ vim simplechain.c
[festerva@hoare7 festerva.1]$ ./a.out 6
i:1 process ID:17504 parent ID:14806 child ID:17505
i:2 process ID:17505 parent ID:1 child ID:17506
i:3 process ID:17506 parent ID:1 child ID:17507
[festerva@hoare7 festerva.1]$ i:4 process ID:17507 parent ID:17506 child ID:1750
8
i:5 process ID:17508 parent ID:17507 child ID:17509
i:6 process ID:17509 parent ID:17508 child ID:0
[festerva@hoare7 festerva.1]$
```

```
[festerva@hoare7 festerva.1]$ ./a.out 1
i:1 process ID:17876 parent ID:14806 child ID:0
[festerva@hoare7 festerva.1]$
```

2.

```
vim simplechain.c
//Fill in the actual process ID's of the processes in the diagram of Figure 8.2 for a run with command line val
of 4

[festerva@hoare7 festerva.1]$ ./a.out 4
i:1 process ID:18430 parent ID:14806 child ID:18431
i:2 process ID:18431 parent ID:1 child ID:18432
i:3 process ID:18432 parent ID:18431 child ID:18433
i:4 process ID:18433 parent ID:18432 child ID:0
```

3. Experiment with different values for the command.

line argument to find largest numbers of processes that the program can generate. Observe the fraction that are adopted by init. */

This one was with max at 100, but I was told in class the max should be 30. Skipped some screenshots in the middle. Seems to be some variation to the order of the messages.

```

[festerva@hoare7 festerva.1]$ i:4 process ID:17507 parent ID:17506 child ID:1750
8
i:5 process ID:17508 parent ID:17507 child ID:17509
i:6 process ID:17509 parent ID:17508 child ID:0
[festerva@hoare7 festerva.1]$ vim simplechain.c
[festerva@hoare7 festerva.1]$ gcc simplechain.c
[festerva@hoare7 festerva.1]$ ./a.out 1
i:1 process ID:17876 parent ID:14806 child ID:0
[festerva@hoare7 festerva.1]$ ./a.out 100
i:1 process ID:17930 parent ID:14806 child ID:17931
i:2 process ID:17931 parent ID:1 child ID:17932
i:3 process ID:17932 parent ID:17931 child ID:17933
i:4 process ID:17933 parent ID:17932 child ID:17934
[festerva@hoare7 festerva.1]$ i:5 process ID:17934 parent ID:1 child ID:17935
i:6 process ID:17935 parent ID:17934 child ID:17936
i:7 process ID:17936 parent ID:17935 child ID:17937
i:8 process ID:17937 parent ID:17936 child ID:17938
i:9 process ID:17938 parent ID:17937 child ID:17939
i:10 process ID:17939 parent ID:17938 child ID:17940
i:11 process ID:17940 parent ID:1 child ID:17941
i:12 process ID:17941 parent ID:17940 child ID:17942
i:13 process ID:17942 parent ID:17941 child ID:17943
i:14 process ID:17943 parent ID:17942 child ID:17944
i:15 process ID:17944 parent ID:17943 child ID:17945
i:16 process ID:17945 parent ID:1 child ID:17946

```

```

festerva@hoare7:~/reposito
i:16 process ID:17945 parent ID:1 child ID:17946
i:17 process ID:17946 parent ID:1 child ID:17947
i:18 process ID:17947 parent ID:17946 child ID:17948
i:19 process ID:17948 parent ID:17947 child ID:17949
i:20 process ID:17949 parent ID:17948 child ID:17950
i:21 process ID:17950 parent ID:17949 child ID:17951
i:22 process ID:17951 parent ID:17950 child ID:17952
i:23 process ID:17952 parent ID:17951 child ID:17953
i:24 process ID:17953 parent ID:17952 child ID:17954
i:25 process ID:17954 parent ID:1 child ID:17955
i:26 process ID:17955 parent ID:17954 child ID:17956
i:27 process ID:17956 parent ID:17955 child ID:17957
i:28 process ID:17957 parent ID:17956 child ID:17958
i:29 process ID:17958 parent ID:1 child ID:17959
i:30 process ID:17959 parent ID:17958 child ID:17960
i:31 process ID:17960 parent ID:17959 child ID:17961
i:32 process ID:17961 parent ID:17960 child ID:17962
i:33 process ID:17962 parent ID:1 child ID:17963
i:34 process ID:17963 parent ID:17962 child ID:17964
i:35 process ID:17964 parent ID:17963 child ID:17965
i:36 process ID:17965 parent ID:1 child ID:17966
i:37 process ID:17966 parent ID:1 child ID:17967
i:38 process ID:17967 parent ID:17966 child ID:17968
i:39 process ID:17968 parent ID:17967 child ID:17969
i:40 process ID:17969 parent ID:1 child ID:17970
i:41 process ID:17970 parent ID:1 child ID:17971

```

```

festerva@hoare7:~
i:76 process ID:18005 parent ID:18004 child ID:18006
i:77 process ID:18006 parent ID:18005 child ID:18007
i:78 process ID:18007 parent ID:18006 child ID:18008
i:79 process ID:18008 parent ID:1 child ID:18009
i:80 process ID:18009 parent ID:1 child ID:18010
i:81 process ID:18010 parent ID:1 child ID:18011
i:82 process ID:18011 parent ID:18010 child ID:18012
i:83 process ID:18012 parent ID:18011 child ID:18013
i:84 process ID:18013 parent ID:1 child ID:18014
i:85 process ID:18014 parent ID:18013 child ID:18015
i:86 process ID:18015 parent ID:1 child ID:18016
i:87 process ID:18016 parent ID:18015 child ID:18017
i:88 process ID:18017 parent ID:18016 child ID:18018
i:89 process ID:18018 parent ID:1 child ID:18019
i:90 process ID:18019 parent ID:1 child ID:18020
i:91 process ID:18020 parent ID:18019 child ID:18021
i:92 process ID:18021 parent ID:18020 child ID:18022
i:93 process ID:18022 parent ID:1 child ID:18023
i:94 process ID:18023 parent ID:18022 child ID:18024
i:95 process ID:18024 parent ID:18023 child ID:18025
i:96 process ID:18025 parent ID:1 child ID:18026
i:97 process ID:18026 parent ID:18025 child ID:18027
i:98 process ID:18027 parent ID:18026 child ID:18028
i:99 process ID:18028 parent ID:1 child ID:18029
i:100 process ID:18029 parent ID:18028 child ID:0

```

4. /* Put sleep[10} directly before the final fprintf statement in program 3.1. What is the max number of processes generated in this process? */

It waited for a while and then did this. I noticed there are two 1s together in the parent ID. The max processes is messed up currently on the server.

```

[festerva@hoare7 festerva.1]$ gcc simplechain2.c
[festerva@hoare7 festerva.1]$ ./a.out 5
i:1 process ID:18921 parent ID:18783 child ID:18922
i:2 process ID:18922 parent ID:1 child ID:18923
i:3 process ID:18923 parent ID:1 child ID:18924
i:4 process ID:18924 parent ID:18923 child ID:18925
[festerva@hoare7 festerva.1]$ i:5 process ID:18925 parent ID:1 child ID:0

```

5.

/*Put loop around the final fprintf in program 3.1. Have the loop execute k times. Put sleep(m); inside the loop after the fprintf.

pass ka nad m on the command line. Run the program for several values of n, k and m and observe the results.

It ran 10 processes for k times. Some of them seem out of order. Perhaps the sleep interrupted the order.


```
festerva@hoare7 festerva.1]$ gcc simplechain3.c
festerva@hoare7 festerva.1]$ ./a.out 10
```

```
Enter value for k: 4
```

```
Enter value for m: 3
```

```
:0 process ID:22298 parent ID:7164 child ID:22313
:0 process ID:22313 parent ID:22298 child ID:22314
:0 process ID:22314 parent ID:22313 child ID:22315
:0 process ID:22315 parent ID:22314 child ID:22316
:0 process ID:22316 parent ID:22315 child ID:22317
:0 process ID:22317 parent ID:22316 child ID:22318
:0 process ID:22318 parent ID:22317 child ID:22319
:0 process ID:22319 parent ID:22318 child ID:22320
:0 process ID:22320 parent ID:22319 child ID:22321
:0 process ID:22321 parent ID:22320 child ID:0
:1 process ID:22298 parent ID:7164 child ID:22313
:1 process ID:22313 parent ID:22298 child ID:22314
:1 process ID:22314 parent ID:22313 child ID:22315
:1 process ID:22315 parent ID:22314 child ID:22316
:1 process ID:22316 parent ID:22315 child ID:22317
:1 process ID:22317 parent ID:22316 child ID:22318
:1 process ID:22318 parent ID:22317 child ID:22319
:1 process ID:22319 parent ID:22318 child ID:22320
```

```
i:1 process ID:22319 parent ID:22318 child ID:22320
i:1 process ID:22321 parent ID:22320 child ID:0
i:1 process ID:22320 parent ID:22319 child ID:22321
i:2 process ID:22298 parent ID:7164 child ID:22313
i:2 process ID:22313 parent ID:22298 child ID:22314
i:2 process ID:22314 parent ID:22313 child ID:22315
i:2 process ID:22315 parent ID:22314 child ID:22316
i:2 process ID:22316 parent ID:22315 child ID:22317
i:2 process ID:22317 parent ID:22316 child ID:22318
i:2 process ID:22318 parent ID:22317 child ID:22319
i:2 process ID:22319 parent ID:22318 child ID:22320
i:2 process ID:22321 parent ID:22320 child ID:0
i:2 process ID:22320 parent ID:22319 child ID:22321
i:3 process ID:22298 parent ID:7164 child ID:22313
i:3 process ID:22313 parent ID:22298 child ID:22314
i:3 process ID:22314 parent ID:22313 child ID:22315
i:3 process ID:22315 parent ID:22314 child ID:22316
i:3 process ID:22316 parent ID:22315 child ID:22317
i:3 process ID:22317 parent ID:22316 child ID:22318
i:3 process ID:22318 parent ID:22317 child ID:22319
i:3 process ID:22319 parent ID:22318 child ID:22320
i:3 process ID:22321 parent ID:22320 child ID:0
i:3 process ID:22320 parent ID:22319 child ID:22321
```

6.

/*Modify program 3.1 by putting a wait function call before the final printf statement. Does this affect the output of the program?*/

It made the processes come out of order because the parent waited for a child.

```
[festerva@hoare7 festerva.1]$ vim simplechain4.c
[festerva@hoare7 festerva.1]$ gcc simplechain4.c
[festerva@hoare7 festerva.1]$ ./a.out 10
i:1 process ID:21328 parent ID:21327 child ID:1
i:2 process ID:21329 parent ID:21327 child ID:1
i:3 process ID:21330 parent ID:21327 child ID:1
i:6 process ID:21333 parent ID:21327 child ID:1
i:5 process ID:21332 parent ID:21327 child ID:1
i:10 process ID:21327 parent ID:7164 child ID:0
i:7 process ID:21334 parent ID:21327 child ID:1
i:9 process ID:21336 parent ID:1 child ID:1
i:8 process ID:21335 parent ID:1 child ID:1
i:4 process ID:21331 parent ID:1 child ID:1
[festerva@hoare7 festerva.1]$
```

7. /*Replace the final fprintf statement with four fprintf statements, one for each four integers displayed.

Only the last one should output a newline. What happens when you run this? Can you tell which process generates each part of the output? Run the program several times and see if there is a difference in output

```
[festerva@hoare7 festerva.1]$ gcc simplechain5.c
[festerva@hoare7 festerva.1]$ ./a.out 7
i:1 processID:19298 processParent Id:18783 process Child ID:19299 process
i:2 processID:19299 processParent Id:1 process Child ID:19300 process
i:3 processID:19300 processParent Id:1 process Child ID:19301 process
i:4 processID:19301 processParent Id:1 process Child ID:19302 process
[festerva@hoare7 festerva.1]$ i:5 processID:19302 processParent Id:1 process Child ID:19303 process
i:7 processID:19304 processID:19303 processParent Id:19303 process Child ID:0 process
Parent Id:1 process Child ID:19304 process
```

```
./a.out 10
i:1 processID:19430 processParent Id:18783 process Child ID:19431 process
i:2 processID:19431 processParent Id:1 process Child ID:19432 process
i:3 processID:19432 processParent Id:1 process Child ID:19433 process
[festerva@hoare7 festerva.1]$ i:4 processID:19433 processParent Id:1 process Child ID:19434 process
i:5 processID:19434 processParent Id:19433 process Child ID:19435 process
i:6 processID:19435 processParent Id:1 process Child ID:19436 process
i:7 processID:19436 processParent Id:1 process Child ID:19437 process
i:8 processID:19437 processParent Id:19436 process Child ID:19438 process
i:9 processID:19438 processID:19439 processParent Id:1 process Child ID:19439 process
ID:19439 processParent Id:19438 process Child ID:0 process
```

Try without labeling what it is.

```
[festerva@hoare7 festerva.1]$ ./a.out 5
126244716426245
226245126246
326246126247
426247126248
52624810
[festerva@hoare7 festerva.1]$
```

Nope, can't tell

/* replace final fprintf statement with a loop that reads nchars characters from a standard input, one character at a time and puts them in an array called mybuf. The values of n and nchars should be passed as command line arguments. After the loop put a '\0' character in entry nchars of the array so that it contains a string. Output to standard error in a single fprintf the process ID followed by the string in mybuff. Run the program for several values of n and nchars. Observe results. Press the return key often and continuing typing at the keyboard until all the processes have exited */


















```
[festerva@hoare7 festerva.1]$ gcc simplechain6.c
[festerva@hoare7 festerva.1]$ ./a.out 3 4
t
h
2
5
26943 process ID: th25
```

```
[festerva@hoare7 festerva.1]$ ./a.out 3 5
4
5
6
7
8
27153 process ID: 45678
[festerva@hoare7 festerva.1]$
```

This is with instructions of continuing typing and return key often. First line is what is expected, but continues to do something after randomly.

```
[festerva@hoare7 festerva.1]$ ./a.out 3 5
5
4
3
6
7
27657 process ID: 54367
[festerva@hoare7 festerva.1]$ 27658 process ID: 55555
27659 process ID:
35
-bash: 35: command not found
```

Git

 rachel festervand finished project		Latest commit 7b68250 11 minutes ago
 .README.md.swp	finished project	11 minutes ago
 Makefile	finished project	11 minutes ago
 Makefile.mak	Add files via upload	5 days ago
 README.md	finished project	11 minutes ago
 Readme.txt	Add files via upload	5 days ago
 a.out	finished project	11 minutes ago
 answers	finished project	11 minutes ago
 problem3.c	finished project	11 minutes ago
 simplechain.c	Add files via upload	2 days ago
 simplechain1.c	finished project	11 minutes ago
 simplechain2.c	Add files via upload	2 days ago
 simplechain3.c	finished project	11 minutes ago
 simplechain4.c	finished project	11 minutes ago
 simplechain5.c	finished project	11 minutes ago
 simplechain6.c	finished project	11 minutes ago
 README.md		