Asa Dillahunty
asdillahunty
11630104
CS300 Programming Project

Initial testing:

```
./searchmanager 2 con pre wor

Message(1): "con" Sent (8 bytes)

Report "con"
Passage 0 - Sense_And_Sensibility.txt - constant
Passage 1 - Mansfield_Park.txt - contemptible
Passage 2 - The_Call_Of_The_Wild.txt - not found
Passage 3 - Tale_Of_Two_Cities.txt - not found
Passage 4 - Peter_Pan.txt - conspicuous

Message(2): "pre" Sent (8 bytes)

Report "pre"
Passage 0 - Sense_And_Sensibility.txt - not found
Passage 1 - Mansfield_Park.txt - predict
Passage 2 - The_Call_Of_The_Wild.txt - not found
Passage 3 - Tale_Of_Two_Cities.txt - preserves
Passage 4 - Peter_Pan.txt - not found

Message(3): "wor" Sent (8 bytes)

Report "wor"
Passage 0 - Sense_And_Sensibility.txt - not found
Passage 1 - Mansfield_Park.txt - worse
Passage 2 - The_Call_Of_The_Wild.txt - not found
Passage 3 - Tale_Of_Two_Cities.txt - worst
Passage 4 - Peter_Pan.txt - not found

Message(0): "   " Sent (8 bytes)

Exiting ...
```

From here you can see the given test cases work and produce correct output for the search manager. This shows it gives the longest word, and when there are two or more of the same length, it prints the one that is last when sorted alphabetically.

```
java -cp . -Djava.library.path=. edu.cs300.PassageProcessor 2>/dev/null
Worker-0 (Sense_And_Sensibility.txt) thread started ...
Worker-2 (The_Call_Of_The_Wild.txt) thread started ...
Worker-1 (Mansfield_Park.txt) thread started ...
Worker-3 (Tale_Of_Two_Cities.txt) thread started ...
Worker-4 (Peter_Pan.txt) thread started ...
**prefix(1) con recieved
Worker-2 1:con ==> not found
Worker-3 1:con ==> not found
msgsnd Reply 2 of 5 on 1:con from The_Call_Of_The_Wild.txt present=0 lw=----(len=4) msglen=144
msgsnd Reply 3 of 5 on 1:con from Tale_Of_Two_Cities.txt present=0 lw=----(len=4) msglen=144
Worker-4 1:con ==> conspicuous
msgsnd Reply 4 of 5 on 1:con from Peter_Pan.txt present=1 lw=conspicuous(len=11) msglen=144
Worker-0 1:con ==> constant
Worker-1 1:con ==> contemptiblemsgsnd Reply 0 of 5 on 1:con from Sense_And_Sensibility.txt present=1 lw=constant(len=8) msglen=144

msgsnd Reply 1 of 5 on 1:con from Mansfield_Park.txt present=1 lw=contemptible(len=12) msglen=144
**prefix(2) pre recieved
Worker-0 2:pre ==> not found
Worker-4 2:pre ==> not found
Worker-3 2:pre ==> preserves
msgsnd Reply 4 of 5 on 2:pre from Peter_Pan.txt present=0 lw=----(len=4) msglen=144
Worker-1 2:pre ==> predictmsgsnd Reply 3 of 5 on 2:pre from Tale_Of_Two_Cities.txt present=1 lw=preserves(len=9) msglen=144

Worker-2 2:pre ==> not found
msgsnd Reply 1 of 5 on 2:pre from Mansfield_Park.txt present=1 lw=predict(len=7) msglen=144
msgsnd Reply 2 of 5 on 2:pre from The_Call_Of_The_Wild.txt present=0 lw=----(len=4) msglen=144
msgsnd Reply 0 of 5 on 2:pre from Sense_And_Sensibility.txt present=0 lw=----(len=4) msglen=144
**prefix(3) wor recieved
Worker-0 3:wor ==> not found
Worker-3 3:wor ==> worst
Worker-2 3:wor ==> not found
Worker-4 3:wor ==> not found
Worker-1 3:wor ==> worsemsgsnd Reply 0 of 5 on 3:wor from Sense_And_Sensibility.txt present=0 lw=----(len=4) msglen=144

msgsnd Reply 3 of 5 on 3:wor from Tale_Of_Two_Cities.txt present=1 lw=worst(len=5) msglen=144
msgsnd Reply 2 of 5 on 3:wor from The_Call_Of_The_Wild.txt present=0 lw=----(len=4) msglen=144
msgsnd Reply 4 of 5 on 3:wor from Peter_Pan.txt present=0 lw=----(len=4) msglen=144
msgsnd Reply 1 of 5 on 3:wor from Mansfield_Park.txt present=1 lw=worse(len=5) msglen=144
**prefix(0)     recieved
Terminating ...
```

Passage Processor's output is out of order, which is something we like to see in multi-threaded output. It also appears some newline characters are being printed apart from their strings, likely because of threads.

First boundary test case: text files that don't exist
When invalid paths are specified in paths.txt, the code prints an error to stderr and continues. It does not make a thread for that passage, and never sends it anything. The output is exactly the same, only a different order, so it is excluded.

Second boundary test case: all text files in passage.txt are invalid

```
java -cp . -Djava.library.path=. edu.cs300.PassageProcessor 2>/dev/null
Terminating ...
```

The Passage Processor halts as soon as it realizes there are no passages to process, sends a message to stderr and Terminates. Note above that it does not read a prefix or send a message to the search manager.

Third boundary test case: some invalid prefixes

```
[asdillahunty@cs426:~/asdillahunty> ./searchmanager 2 con ke0 jo john
 "ke0" is an invalid prefix.
 "jo" is an invalid prefix.

Message(1): "con" Sent (8 bytes)

Report "con"
Passage 0 - Sense_And_Sensibility.txt - constant
Passage 1 - Mansfield_Park.txt - contemptible
Passage 2 - The_Call_Of_The_Wild.txt - not found
Passage 3 - Tale_Of_Two_Cities.txt - not found
Passage 4 - Peter_Pan.txt - conspicuous

Message(2): "john" Sent (9 bytes)

Report "john"
Passage 0 - Sense_And_Sensibility.txt - not found
Passage 1 - Mansfield_Park.txt - not found
Passage 2 - The_Call_Of_The_Wild.txt - not found
Passage 3 - Tale_Of_Two_Cities.txt - not found
Passage 4 - Peter_Pan.txt - not found

Message(0): "   " Sent (8 bytes)

Exiting ...
```

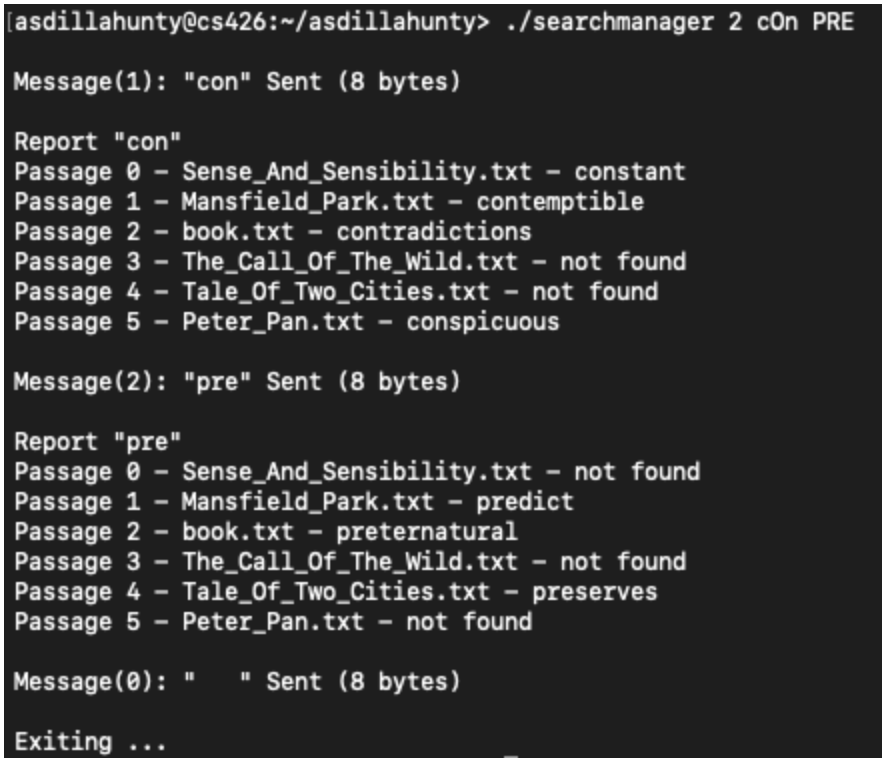
As you can see above, ke0 and jo are invalid prefixes. I have in the code “fprintf(stderr, “message”), but it insists on showing up in the standard output. This demonstrates words with nonalphabetic characters are ignored, and the program continues if only some prefixes are invalid.

Fourth boundary test case: all invalid prefixes

```
[asdillahunty@cs426:~/asdillahunty> ./searchmanager 2 qwertyuiopasdfghjklzxcvb
 "qwertyuiopasdfghjklzxcvb" is an invalid prefix.
No valid prefixes.
Exiting ...
```

If all prefixes are invalid, it prints that to stderr and exits before getting the chance to send an IPCS message. This sample also illustrates prefixes over 20 letters in length are invalid as well.

Fifth boundary test case: capital prefixes



All prefixes are converted to lowercase at the start of the program.

SIGINT Handler:

```
./searchmanager 2 con pre wor

Message(1): "con" Sent (8 bytes)

^Ccon - 5 of 6
pre - pending
wor - pending
Report "con"
Passage 0 - Sense_And_Sensibility.txt - constant
Passage 1 - Mansfield_Park.txt - contemptible
Passage 2 - book.txt - contradictions
Passage 3 - The_Call_Of_The_Wild.txt - not found
Passage 4 - Tale_Of_Two_Cities.txt - not found
Passage 5 - Peter_Pan.txt - conspicuous

Message(2): "pre" Sent (8 bytes)

^Ccon - done
pre - pending
wor - pending
Report "pre"
Passage 0 - Sense_And_Sensibility.txt - not found
Passage 1 - Mansfield_Park.txt - predict
Passage 2 - book.txt - preternatural
Passage 3 - The_Call_Of_The_Wild.txt - not found
Passage 4 - Tale_Of_Two_Cities.txt - preserves
Passage 5 - Peter_Pan.txt - not found

Message(3): "wor" Sent (8 bytes)

Report "wor"
Passage 0 - Sense_And_Sensibility.txt - not found
Passage 1 - Mansfield_Park.txt - worse
Passage 2 - book.txt - worthlessness
Passage 3 - The_Call_Of_The_Wild.txt - not found
Passage 4 - Tale_Of_Two_Cities.txt - worst
Passage 5 - Peter_Pan.txt - not found

Message(0): "   " Sent (8 bytes)

Exiting ...
```

This is illustrating the use of the signal interrupt. It was very difficult to get a message of the format: “%d of %d”. To do so I had to use a massive text document to slow down the building of the Trie. It’s next to impossible to catch it in the middle without using sleep statements. But as you can see it does work as expected, printing pending for prefixes not started, how many completed out of how many total for those in progress, and done for finished prefixes.