

# Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester IV

2020

---

## UCS1411 - Operating Systems Laboratory

---

### Exercise – 2- Simulation of system commands using system calls

#### *Objective:*

To develop a C program to implement the cp, ls, grep commands (with some options) using system calls.

#### *Code:*

Q1. To develop a C program to implement the cp command using system calls

```
1 //Implementing cp command
2 #include<stdio.h>
3 #include<stdlib.h>
4 #include<sys/types.h>
5 #include<fcntl.h>
6 #include<string.h>
7 #include<unistd.h>
8
```

```

9 void main (int argc, char *argv[]){
10     if(argc<3)
11         printf("\n Insufficient arguments \n");
12     else{
13         if(argc==3){
14             //Non-interactive
15             int sourcefd=open(argv[1],O_RDWR);
16             //Non-existent source file
17             if(sourcefd==-1){
18                 printf("\n Source file does not exist \n");
19             }
20             else{
21                 //Reading source file
22                 char *tmpline=(char*)calloc(1000,sizeof(char)
23 );
24
25                 int readfd=read(sourcefd,tmpline,100);
26
27                 tmpline[readfd]='\0';
28
29                 //Creating destination file if non-existent
30                 int destfd=open(argv[2],O_CREAT|O_RDWR);
31                 if(destfd==-1)
32                     printf("\n Destination file could not be
33 created \n");
34                 else{
35                     write(destfd,tmpline,strlen(tmpline));
36                     printf("\n Content copied successfully\n"
37 );
38                     close(destfd);
39                 }
40                 close(sourcefd);
41             }
42         }
43         else{
44             //Interactive
45             int sourcefd=open(argv[2],O_RDWR);
46             //Non-existent source file
47             if(sourcefd==-1){
48                 printf("\n Source file does not exist \n");
49             }
50             else{
51                 //Reading source file
52                 char *tmpline=(char*)calloc(1000,sizeof(char)
53 );
54
55                 int readfd=read(sourcefd,tmpline,100);
56
57                 tmpline[readfd]='\0';
58
59                 //Creating destination file if non-existent

```

```

54         int destfd=open(argv[3],O_CREAT|O_RDWR);
55         if(destfd==-1)
56             printf("\n Destination file could not be
created \n");
57         else{
58             char opt;
59             if(strcmp(argv[1],"-i")==0){
60                 printf("\n Copy contents? y/n ");
61                 scanf(" %c",&opt);
62                 if(opt=='Y' || opt=='y'){
63                     write(destfd,tmpline,strlen(
tmpline));
64                     printf("\n Content copied
successfully\n");
65                 }
66                 else{
67                     printf("\n Manual Abort. \n");
68                 }
69             }
70             else{
71                 write(destfd,tmpline,strlen(tmpline))
;
72                 printf("\n Content copied
successfully\n");
73             }
74             close(destfd);
75         }
76         close(sourcefd);
77     }
78 }
79 }
80 }

```

### ***Output:***

```

1 Output:
2 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
SystemCalls$ ./mycp
3
4 Insufficient arguments
5 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
SystemCalls$ ./mycp -i source.txt
6
7 Source file does not exist
8 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
SystemCalls$ ./mycp -i source.txt dest.txt
9
10 Copy contents? y/n y
11

```

```

12 Content copied successfully
13 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
   SystemCalls$ ./mycp source.txt dest.txt
14
15 Content copied successfully
16 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
   SystemCalls$ cat source.txt
17 shiva
18 sharvan
19 shashuuuu
20 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
   SystemCalls$ cat dest.txt
21 shiva
22 sharvan
23 shashuuuu

```

---

Q2. To develop a C program to implement the ls command using system calls

```

1 //Implementing ls command
2 #include<stdio.h>
3 #include<stdlib.h>
4 #include<sys/types.h>
5 #include<sys/stat.h>
6 #include<fcntl.h>
7 #include<string.h>
8 #include<unistd.h>
9 #include<dirent.h>
10
11 void Recurse_ls(char *dirname){
12     DIR *dirstream=NULL;
13     struct dirent *dirpointer=NULL;
14
15     if(dirname[0]=='.'){
16         getcwd(dirname,1000);
17     }
18     dirstream=opendir(dirname);
19     if(dirstream==NULL){
20         printf("\n Unable to open directory \n");
21     }
22     else{
23         struct stat dirstat;
24         while(NULL != (dirpointer=readdir(dirstream))){
25             printf("%s\t",dirpointer->d_name);
26         }
27         printf("\n\n");
28         dirpointer=NULL;

```

```

29         while(NULL != (dirpointer=readdir(dirstream))){
30             int dirfd=stat(dirpointer->d_name,&dirstat);
31             if(dirfd<0)
32                 printf("\nUnable to locate file\n");
33             else{
34                 int check_dir=S_ISDIR(dirstat.st_mode);
35                 if(check_dir){
36                     Recurse_ls(dirpointer->d_name);
37                 }
38             }
39         }
40     }
41 }
42 void main (int argc,char *argv[]){
43
44     char *dirname=(char*)calloc(1000,sizeof(char));
45
46     printf("\nEnter path of directory: ");
47     scanf(" %s",dirname);
48     if(argc==1){
49         DIR *dirstream=NULL;
50         struct dirent *dirpointer=NULL;
51
52         if(dirname[0]=='.'){
53             getcwd(dirname,1000);
54         }
55         dirstream=opendir(dirname);
56         if(dirstream==NULL){
57             printf("\n Unable to open directory \n");
58         }
59         else{
60             while(NULL != (dirpointer=readdir(dirstream))){
61                 printf("\n %s",dirpointer->d_name);
62             }
63             printf("\n");
64
65             closedir(dirstream);
66         }
67     }
68     else if(strcmp(argv[1],"-l")==0){
69         DIR *dirstream=NULL;
70         struct dirent *dirpointer=NULL;
71
72         if(dirname[0]=='.'){
73             getcwd(dirname,1000);
74         }
75         dirstream=opendir(dirname);
76         if(dirstream==NULL){
77             printf("\n Unable to open directory \n");

```

```

78     }
79     else{
80         struct stat dirstat;
81         while(NULL != (dirpointer=readdir(dirstream))){
82
83             int dirfd=stat(dirpointer->d_name,&dirstat);
84             if(dirfd<0)
85                 printf("\nUnable to locate directory\n");
86             else{
87                 printf((S_ISDIR(dirstat.st_mode))?"d":"-")
88 );
89                 printf( (dirstat.st_mode & S_IRUSR) ? "r"
90 : "-");
91                 printf( (dirstat.st_mode & S_IWUSR) ? "w"
92 : "-");
93                 printf( (dirstat.st_mode & S_IXUSR) ? "x"
94 : "-");
95                 printf( (dirstat.st_mode & S_IRGRP) ? "r"
96 : "-");
97                 printf( (dirstat.st_mode & S_IWGRP) ? "w"
98 : "-");
99                 printf( (dirstat.st_mode & S_IXGRP) ? "x"
100 : "-");
101                 printf( (dirstat.st_mode & S_IROTH) ? "r"
102 : "-");
103                 printf( (dirstat.st_mode & S_IWOTH) ? "w"
104 : "-");
105                 printf(" %ld ",dirstat.st_nlink);
106                 printf(" %6ld ",dirstat.st_size);
107                 printf(" %8ld ",dirstat.st_ino);
108                 printf(" %15s",dirpointer->d_name);
109             }
110             printf("\n");
111         }
112     }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

### Output:

```

1 Output:
2 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
  SystemCalls$ ./mys
3

```

```

4 Enter path of directory: .
5
6 mycp.c
7 2-ex2-cmds.pdf
8 mygrep.c
9 .
10 dest.txt
11 CPUScheduling
12 SYSTEM_CALLS_MANUAL.pdf
13 a
14 grep.c
15 mycp
16 myls
17 ls
18 mygrep
19 myls.c
20 source.txt
21 ..
22 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
   SystemCalls$ ./mys -l
23
24 Enter path of directory: .
25 -rw-rw--r- 1      2625    2363958          mycp.c
26 -rw-rw--r- 1    44937    3670881    2-ex2-cmds.pdf
27 -rw-r---r- 1     1358    2363973          mygrep.c
28 drwxr-xr- 3     4096    2363810          .
29 -r--rw-xr- 1        24    2360766          dest.txt
30 drwxr-xr- 2     4096    2368128    CPUScheduling
31 -rw-rw--r- 1    19896    3670880    SYSTEM_CALLS_MANUAL.pdf
32 -rwxrwxr- 1    12920    2360763          a
33 -rw-r---r- 1     2506    2360758          grep.c
34 -rwxrwxr- 1    12840    2360764          mycp
35 -rwxrwxr- 1    12992    2360767          myls
36 -rwxrwxr- 1    12992    2368119          ls
37 -rwxrwxr- 1     8560    2363966          mygrep
38 -rw-r---r- 1     2864    2363972          myls.c
39 -rw-rw--r- 1        24    2363959          source.txt
40 drwxr-xr- 6     4096    2363963          ..
41 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
   SystemCalls$ ./mys -R
42
43 Enter path of directory: .
44 mycp.c 2-ex2-cmds.pdf mygrep.c . dest.txt
   CPUScheduling SYSTEM_CALLS_MANUAL.pdf a grep.c mycp
   myls ls mygrep myls.c source.txt ..

```

---

Q3. To develop a C program to implement the grep command using system

calls

```
1 //Implementing grep command
2 #include<stdio.h>
3 #include<stdlib.h>
4 #include<sys/types.h>
5 #include<fcntl.h>
6 #include<string.h>
7 #include<unistd.h>
8
9 int isSubstring(char* s1, char* s2)
10 {
11     int M = strlen(s1);
12     int N = strlen(s2);
13
14     //Loop to slide pat[] one by one
15     for (int i = 0; i <= N - M; i++) {
16         int j;
17
18         // For current index i, check for pattern match
19         for (j = 0; j < M; j++)
20             if (s2[i + j] != s1[j])
21                 break;
22
23         if (j == M)
24             return i;
25     }
26
27     return -1;
28 }
29
30 void main (int argc, char *argv[]){
31     if(argc<2)
32         printf("\n Insufficient arguments \n");
33     else{
34
35         int filefd;
36         //printf("\n%d\n", argc);
37
38         if(argc==3)
39             filefd=open(argv[2], O_RDWR);
40         else
41             filefd=open(argv[3], O_RDWR);
42
43         //Non-existent source file
44         if(filefd==-1){
45             printf("\n Source file does not exist \n");
46         }
47         else{
48             //Reading source file
```



```

49     char *tmpline=(char*)calloc(1000,sizeof(char));
50     int readfd=read(filefd,tmpline,100);
51
52     tmpline[readfd]='\0';
53
54     char *lines[100];
55     for(int i=0;i<100;i++){
56         lines[i]=(char*)malloc(sizeof(100));
57     }
58     int lctr=0;
59     char* token=strtok(tmpline,"\n");
60     while(token!=NULL){
61         strcpy(lines[lctr++],token);
62         token=strtok(NULL,"\n");
63     }
64
65     if(argc==3){
66         for(int i=0;i<lctr;i++){
67             if(isSubstring(argv[1],lines[i])!=-1)
68                 printf("\n%s\n",lines[i]);
69         }
70     }
71     else{
72         if(strcmp(argv[1],"-c")==0){
73             int ctr=0;
74             for(int i=0;i<lctr;i++){
75                 if(isSubstring(argv[2],lines[i])!=-1)
76                     ctr++;
77             }
78             printf("\n%d\n",ctr);
79         }
80         else if(strcmp(argv[1],"-v")==0){
81             for(int i=0;i<lctr;i++){
82                 if(isSubstring(argv[2],lines[i])==1)
83                     printf("\n%s\n",lines[i]);
84             }
85         }
86         else if(strcmp(argv[1],"-n")==0){
87             int ctr=0;
88             for(int i=0;i<lctr;i++){
89                 if(isSubstring(argv[2],lines[i])!=-1)
90                     ctr++;
91                 printf("\n%d %s\n",ctr,lines[i]);
92             }
93         }
94     }
95 }
96

```

```
97         close(filefd);
98     }
99 }
100 }
```

***Output:***

```
1 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
  SystemCalls$ ./a hi source.txt
2
3 shiva
4 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
  SystemCalls$ ./a -c hi source.txt
5
6 1
7 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
  SystemCalls$ ./a -v hi source.txt
8
9 sharvan
10
11 shashuuuu
12 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
  SystemCalls$ ./a -n hi source.txt
13
14 1 shiva
15 shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/
  SystemCalls$ cat source.txt
16 shiva
17 sharvan
18 shashuuuu
```

---