

Лабораторная работа №11

**Программирование в командном процессоре ОС UNIX. Ветвления и
циклы**

Сиссе Мохамед Ламин; НММбд-01-22

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	15
6	Контрольные вопросы	16
	Список литературы	20

Список иллюстраций

4.1	Первая программа	9
4.2	Вызов программы в терминале	10
4.3	Результат	10
4.4	Результат	11
4.5	Вторая программа	11
4.6	Вторая программа	12
4.7	Результат	12
4.8	Третья программа	13
4.9	Результат	13
4.10	Четвертая программа	14
4.11	Вызов программы в терминале	14

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:
 - `-iinputfile` — прочитать данные из указанного файла;
 - `-ooutputfile` — вывести данные в указанный файл;
 - `-р` — указать шаблон для поиска;
 - `-C` — различать большие и малые буквы;
 - `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например `1.tmp`, `2.tmp`, `3.tmp`, `4.tmp` и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

4. Написать командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`).

3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

- С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

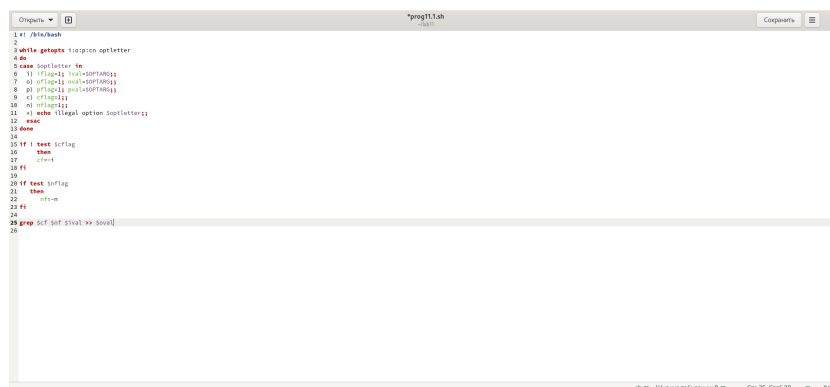
POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны

на базе оболочки Korn. Рассмотрим основные элементы программирования в оболочке `bash`. В других оболочках большинство команд будет совпадать с описанными ниже. [**Prog:bash?**]

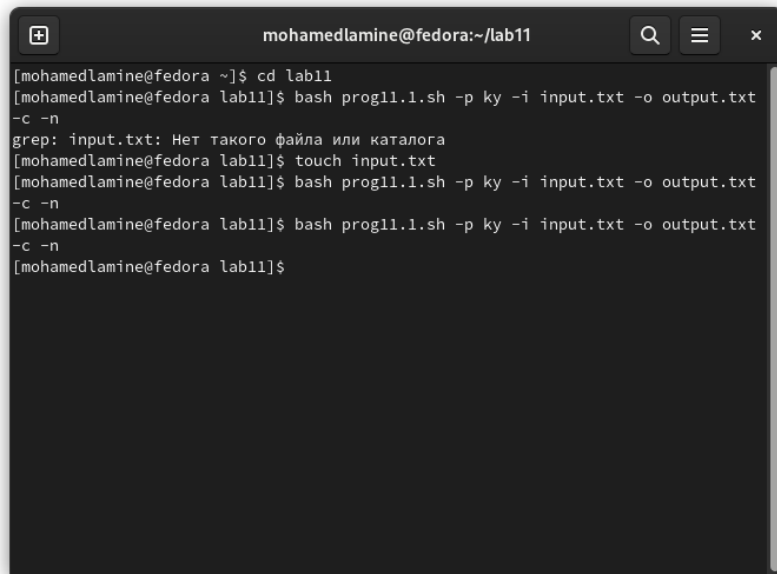
4 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-ршаблон` — указать шаблон для поиска;
- `-С` — различать большие и малые буквы;
- `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-р`. (рис. [4.1])



```
1 #!/bin/bash
2
3 while getopts i:op:C:n: prog11.sh
4 do
5     case $OPTARG in
6         i) iflag=$OPTARG;;
7         o) oflag=$OPTARG;;
8         p) pflag=$OPTARG;;
9         C) iflag=$OPTARG;;
10        n) nflag=$OPTARG;;
11        *) echo "Usage: prog11.sh [-iinputfile] [-ooutputfile] [-pшаблон] [-C] [-n]"; exit 1;;
12    esac
13 done
14
15 if [ -n "$iflag" ]; then
16     iflag=$iflag
17 else
18     iflag=""
19 fi
20
21 if [ -n "$oflag" ]; then
22     oflag=$oflag
23 else
24     oflag=""
25 fi
26
27 if [ -n "$pflag" ]; then
28     pflag=$pflag
29 else
30     pflag=""
31 fi
32
33 if [ -n "$nflag" ]; then
34     nflag=$nflag
35 else
36     nflag=""
37 fi
38
39 if [ -n "$nflag" ]; then
40     nflag=$nflag
41 else
42     nflag=""
43 fi
44
45 if [ -n "$pflag" ]; then
46     pflag=$pflag
47 else
48     pflag=""
49 fi
50
51 if [ -n "$nflag" ]; then
52     nflag=$nflag
53 else
54     nflag=""
55 fi
56
57 if [ -n "$pflag" ]; then
58     pflag=$pflag
59 else
60     pflag=""
61 fi
62
63 if [ -n "$nflag" ]; then
64     nflag=$nflag
65 else
66     nflag=""
67 fi
68
69 if [ -n "$pflag" ]; then
70     pflag=$pflag
71 else
72     pflag=""
73 fi
74
75 if [ -n "$nflag" ]; then
76     nflag=$nflag
77 else
78     nflag=""
79 fi
80
81 if [ -n "$pflag" ]; then
82     pflag=$pflag
83 else
84     pflag=""
85 fi
86
87 if [ -n "$nflag" ]; then
88     nflag=$nflag
89 else
90     nflag=""
91 fi
92
93 if [ -n "$pflag" ]; then
94     pflag=$pflag
95 else
96     pflag=""
97 fi
98
99 if [ -n "$nflag" ]; then
100    nflag=$nflag
101 else
102    nflag=""
103 fi
104
105 if [ -n "$pflag" ]; then
106    pflag=$pflag
107 else
108    pflag=""
109 fi
110
111 if [ -n "$nflag" ]; then
112    nflag=$nflag
113 else
114    nflag=""
115 fi
116
117 if [ -n "$pflag" ]; then
118    pflag=$pflag
119 else
120    pflag=""
121 fi
122
123 if [ -n "$nflag" ]; then
124    nflag=$nflag
125 else
126    nflag=""
127 fi
128
129 if [ -n "$pflag" ]; then
130    pflag=$pflag
131 else
132    pflag=""
133 fi
134
135 if [ -n "$nflag" ]; then
136    nflag=$nflag
137 else
138    nflag=""
139 fi
140
141 if [ -n "$pflag" ]; then
142    pflag=$pflag
143 else
144    pflag=""
145 fi
146
147 if [ -n "$nflag" ]; then
148    nflag=$nflag
149 else
150    nflag=""
151 fi
152
153 if [ -n "$pflag" ]; then
154    pflag=$pflag
155 else
156    pflag=""
157 fi
158
159 if [ -n "$nflag" ]; then
160    nflag=$nflag
161 else
162    nflag=""
163 fi
164
165 if [ -n "$pflag" ]; then
166    pflag=$pflag
167 else
168    pflag=""
169 fi
170
171 if [ -n "$nflag" ]; then
172    nflag=$nflag
173 else
174    nflag=""
175 fi
176
177 if [ -n "$pflag" ]; then
178    pflag=$pflag
179 else
180    pflag=""
181 fi
182
183 if [ -n "$nflag" ]; then
184    nflag=$nflag
185 else
186    nflag=""
187 fi
188
189 if [ -n "$pflag" ]; then
190    pflag=$pflag
191 else
192    pflag=""
193 fi
194
195 if [ -n "$nflag" ]; then
196    nflag=$nflag
197 else
198    nflag=""
199 fi
200
201 if [ -n "$pflag" ]; then
202    pflag=$pflag
203 else
204    pflag=""
205 fi
206
207 if [ -n "$nflag" ]; then
208    nflag=$nflag
209 else
210    nflag=""
211 fi
212
213 if [ -n "$pflag" ]; then
214    pflag=$pflag
215 else
216    pflag=""
217 fi
218
219 if [ -n "$nflag" ]; then
220    nflag=$nflag
221 else
222    nflag=""
223 fi
224
225 if [ -n "$pflag" ]; then
226    pflag=$pflag
227 else
228    pflag=""
229 fi
230
231 if [ -n "$nflag" ]; then
232    nflag=$nflag
233 else
234    nflag=""
235 fi
236
237 if [ -n "$pflag" ]; then
238    pflag=$pflag
239 else
240    pflag=""
241 fi
242
243 if [ -n "$nflag" ]; then
244    nflag=$nflag
245 else
246    nflag=""
247 fi
248
249 if [ -n "$pflag" ]; then
250    pflag=$pflag
251 else
252    pflag=""
253 fi
254
255 if [ -n "$nflag" ]; then
256    nflag=$nflag
257 else
258    nflag=""
259 fi
260
261 if [ -n "$pflag" ]; then
262    pflag=$pflag
263 else
264    pflag=""
265 fi
266
267 if [ -n "$nflag" ]; then
268    nflag=$nflag
269 else
270    nflag=""
271 fi
272
273 if [ -n "$pflag" ]; then
274    pflag=$pflag
275 else
276    pflag=""
277 fi
278
279 if [ -n "$nflag" ]; then
280    nflag=$nflag
281 else
282    nflag=""
283 fi
284
285 if [ -n "$pflag" ]; then
286    pflag=$pflag
287 else
288    pflag=""
289 fi
290
291 if [ -n "$nflag" ]; then
292    nflag=$nflag
293 else
294    nflag=""
295 fi
296
297 if [ -n "$pflag" ]; then
298    pflag=$pflag
299 else
300    pflag=""
301 fi
302
303 if [ -n "$nflag" ]; then
304    nflag=$nflag
305 else
306    nflag=""
307 fi
308
309 if [ -n "$pflag" ]; then
310    pflag=$pflag
311 else
312    pflag=""
313 fi
314
315 if [ -n "$nflag" ]; then
316    nflag=$nflag
317 else
318    nflag=""
319 fi
320
321 if [ -n "$pflag" ]; then
322    pflag=$pflag
323 else
324    pflag=""
325 fi
326
327 if [ -n "$nflag" ]; then
328    nflag=$nflag
329 else
330    nflag=""
331 fi
332
333 if [ -n "$pflag" ]; then
334    pflag=$pflag
335 else
336    pflag=""
337 fi
338
339 if [ -n "$nflag" ]; then
340    nflag=$nflag
341 else
342    nflag=""
343 fi
344
345 if [ -n "$pflag" ]; then
346    pflag=$pflag
347 else
348    pflag=""
349 fi
350
351 if [ -n "$nflag" ]; then
352    nflag=$nflag
353 else
354    nflag=""
355 fi
356
357 if [ -n "$pflag" ]; then
358    pflag=$pflag
359 else
360    pflag=""
361 fi
362
363 if [ -n "$nflag" ]; then
364    nflag=$nflag
365 else
366    nflag=""
367 fi
368
369 if [ -n "$pflag" ]; then
370    pflag=$pflag
371 else
372    pflag=""
373 fi
374
375 if [ -n "$nflag" ]; then
376    nflag=$nflag
377 else
378    nflag=""
379 fi
380
381 if [ -n "$pflag" ]; then
382    pflag=$pflag
383 else
384    pflag=""
385 fi
386
387 if [ -n "$nflag" ]; then
388    nflag=$nflag
389 else
390    nflag=""
391 fi
392
393 if [ -n "$pflag" ]; then
394    pflag=$pflag
395 else
396    pflag=""
397 fi
398
399 if [ -n "$nflag" ]; then
400    nflag=$nflag
401 else
402    nflag=""
403 fi
404
405 if [ -n "$pflag" ]; then
406    pflag=$pflag
407 else
408    pflag=""
409 fi
410
411 if [ -n "$nflag" ]; then
412    nflag=$nflag
413 else
414    nflag=""
415 fi
416
417 if [ -n "$pflag" ]; then
418    pflag=$pflag
419 else
420    pflag=""
421 fi
422
423 if [ -n "$nflag" ]; then
424    nflag=$nflag
425 else
426    nflag=""
427 fi
428
429 if [ -n "$pflag" ]; then
430    pflag=$pflag
431 else
432    pflag=""
433 fi
434
435 if [ -n "$nflag" ]; then
436    nflag=$nflag
437 else
438    nflag=""
439 fi
440
441 if [ -n "$pflag" ]; then
442    pflag=$pflag
443 else
444    pflag=""
445 fi
446
447 if [ -n "$nflag" ]; then
448    nflag=$nflag
449 else
450    nflag=""
451 fi
452
453 if [ -n "$pflag" ]; then
454    pflag=$pflag
455 else
456    pflag=""
457 fi
458
459 if [ -n "$nflag" ]; then
460    nflag=$nflag
461 else
462    nflag=""
463 fi
464
465 if [ -n "$pflag" ]; then
466    pflag=$pflag
467 else
468    pflag=""
469 fi
470
471 if [ -n "$nflag" ]; then
472    nflag=$nflag
473 else
474    nflag=""
475 fi
476
477 if [ -n "$pflag" ]; then
478    pflag=$pflag
479 else
480    pflag=""
481 fi
482
483 if [ -n "$nflag" ]; then
484    nflag=$nflag
485 else
486    nflag=""
487 fi
488
489 if [ -n "$pflag" ]; then
490    pflag=$pflag
491 else
492    pflag=""
493 fi
494
495 if [ -n "$nflag" ]; then
496    nflag=$nflag
497 else
498    nflag=""
499 fi
500
501 if [ -n "$pflag" ]; then
502    pflag=$pflag
503 else
504    pflag=""
505 fi
506
507 if [ -n "$nflag" ]; then
508    nflag=$nflag
509 else
510    nflag=""
511 fi
512
513 if [ -n "$pflag" ]; then
514    pflag=$pflag
515 else
516    pflag=""
517 fi
518
519 if [ -n "$nflag" ]; then
520    nflag=$nflag
521 else
522    nflag=""
523 fi
524
525 if [ -n "$pflag" ]; then
526    pflag=$pflag
527 else
528    pflag=""
529 fi
530
531 if [ -n "$nflag" ]; then
532    nflag=$nflag
533 else
534    nflag=""
535 fi
536
537 if [ -n "$pflag" ]; then
538    pflag=$pflag
539 else
540    pflag=""
541 fi
542
543 if [ -n "$nflag" ]; then
544    nflag=$nflag
545 else
546    nflag=""
547 fi
548
549 if [ -n "$pflag" ]; then
550    pflag=$pflag
551 else
552    pflag=""
553 fi
554
555 if [ -n "$nflag" ]; then
556    nflag=$nflag
557 else
558    nflag=""
559 fi
560
561 if [ -n "$pflag" ]; then
562    pflag=$pflag
563 else
564    pflag=""
565 fi
566
567 if [ -n "$nflag" ]; then
568    nflag=$nflag
569 else
570    nflag=""
571 fi
572
573 if [ -n "$pflag" ]; then
574    pflag=$pflag
575 else
576    pflag=""
577 fi
578
579 if [ -n "$nflag" ]; then
580    nflag=$nflag
581 else
582    nflag=""
583 fi
584
585 if [ -n "$pflag" ]; then
586    pflag=$pflag
587 else
588    pflag=""
589 fi
590
591 if [ -n "$nflag" ]; then
592    nflag=$nflag
593 else
594    nflag=""
595 fi
596
597 if [ -n "$pflag" ]; then
598    pflag=$pflag
599 else
600    pflag=""
601 fi
602
603 if [ -n "$nflag" ]; then
604    nflag=$nflag
605 else
606    nflag=""
607 fi
608
609 if [ -n "$pflag" ]; then
610    pflag=$pflag
611 else
612    pflag=""
613 fi
614
615 if [ -n "$nflag" ]; then
616    nflag=$nflag
617 else
618    nflag=""
619 fi
620
621 if [ -n "$pflag" ]; then
622    pflag=$pflag
623 else
624    pflag=""
625 fi
626
627 if [ -n "$nflag" ]; then
628    nflag=$nflag
629 else
630    nflag=""
631 fi
632
633 if [ -n "$pflag" ]; then
634    pflag=$pflag
635 else
636    pflag=""
637 fi
638
639 if [ -n "$nflag" ]; then
640    nflag=$nflag
641 else
642    nflag=""
643 fi
644
645 if [ -n "$pflag" ]; then
646    pflag=$pflag
647 else
648    pflag=""
649 fi
650
651 if [ -n "$nflag" ]; then
652    nflag=$nflag
653 else
654    nflag=""
655 fi
656
657 if [ -n "$pflag" ]; then
658    pflag=$pflag
659 else
660    pflag=""
661 fi
662
663 if [ -n "$nflag" ]; then
664    nflag=$nflag
665 else
666    nflag=""
667 fi
668
669 if [ -n "$pflag" ]; then
670    pflag=$pflag
671 else
672    pflag=""
673 fi
674
675 if [ -n "$nflag" ]; then
676    nflag=$nflag
677 else
678    nflag=""
679 fi
680
681 if [ -n "$pflag" ]; then
682    pflag=$pflag
683 else
684    pflag=""
685 fi
686
687 if [ -n "$nflag" ]; then
688    nflag=$nflag
689 else
690    nflag=""
691 fi
692
693 if [ -n "$pflag" ]; then
694    pflag=$pflag
695 else
696    pflag=""
697 fi
698
699 if [ -n "$nflag" ]; then
700    nflag=$nflag
701 else
702    nflag=""
703 fi
704
705 if [ -n "$pflag" ]; then
706    pflag=$pflag
707 else
708    pflag=""
709 fi
710
711 if [ -n "$nflag" ]; then
712    nflag=$nflag
713 else
714    nflag=""
715 fi
716
717 if [ -n "$pflag" ]; then
718    pflag=$pflag
719 else
720    pflag=""
721 fi
722
723 if [ -n "$nflag" ]; then
724    nflag=$nflag
725 else
726    nflag=""
727 fi
728
729 if [ -n "$pflag" ]; then
730    pflag=$pflag
731 else
732    pflag=""
733 fi
734
735 if [ -n "$nflag" ]; then
736    nflag=$nflag
737 else
738    nflag=""
739 fi
740
741 if [ -n "$pflag" ]; then
742    pflag=$pflag
743 else
744    pflag=""
745 fi
746
747 if [ -n "$nflag" ]; then
748    nflag=$nflag
749 else
750    nflag=""
751 fi
752
753 if [ -n "$pflag" ]; then
754    pflag=$pflag
755 else
756    pflag=""
757 fi
758
759 if [ -n "$nflag" ]; then
760    nflag=$nflag
761 else
762    nflag=""
763 fi
764
765 if [ -n "$pflag" ]; then
766    pflag=$pflag
767 else
768    pflag=""
769 fi
770
771 if [ -n "$nflag" ]; then
772    nflag=$nflag
773 else
774    nflag=""
775 fi
776
777 if [ -n "$pflag" ]; then
778    pflag=$pflag
779 else
780    pflag=""
781 fi
782
783 if [ -n "$nflag" ]; then
784    nflag=$nflag
785 else
786    nflag=""
787 fi
788
789 if [ -n "$pflag" ]; then
790    pflag=$pflag
791 else
792    pflag=""
793 fi
794
795 if [ -n "$nflag" ]; then
796    nflag=$nflag
797 else
798    nflag=""
799 fi
800
801 if [ -n "$pflag" ]; then
802    pflag=$pflag
803 else
804    pflag=""
805 fi
806
807 if [ -n "$nflag" ]; then
808    nflag=$nflag
809 else
810    nflag=""
811 fi
812
813 if [ -n "$pflag" ]; then
814    pflag=$pflag
815 else
816    pflag=""
817 fi
818
819 if [ -n "$nflag" ]; then
820    nflag=$nflag
821 else
822    nflag=""
823 fi
824
825 if [ -n "$pflag" ]; then
826    pflag=$pflag
827 else
828    pflag=""
829 fi
830
831 if [ -n "$nflag" ]; then
832    nflag=$nflag
833 else
834    nflag=""
835 fi
836
837 if [ -n "$pflag" ]; then
838    pflag=$pflag
839 else
840    pflag=""
841 fi
842
843 if [ -n "$nflag" ]; then
844    nflag=$nflag
845 else
846    nflag=""
847 fi
848
849 if [ -n "$pflag" ]; then
850    pflag=$pflag
851 else
852    pflag=""
853 fi
854
855 if [ -n "$nflag" ]; then
856    nflag=$nflag
857 else
858    nflag=""
859 fi
860
861 if [ -n "$pflag" ]; then
862    pflag=$pflag
863 else
864    pflag=""
865 fi
866
867 if [ -n "$nflag" ]; then
868    nflag=$nflag
869 else
870    nflag=""
871 fi
872
873 if [ -n "$pflag" ]; then
874    pflag=$pflag
875 else
876    pflag=""
877 fi
878
879 if [ -n "$nflag" ]; then
880    nflag=$nflag
881 else
882    nflag=""
883 fi
884
885 if [ -n "$pflag" ]; then
886    pflag=$pflag
887 else
888    pflag=""
889 fi
890
891 if [ -n "$nflag" ]; then
892    nflag=$nflag
893 else
894    nflag=""
895 fi
896
897 if [ -n "$pflag" ]; then
898    pflag=$pflag
899 else
900    pflag=""
901 fi
902
903 if [ -n "$nflag" ]; then
904    nflag=$nflag
905 else
906    nflag=""
907 fi
908
909 if [ -n "$pflag" ]; then
910    pflag=$pflag
911 else
912    pflag=""
913 fi
914
915 if [ -n "$nflag" ]; then
916    nflag=$nflag
917 else
918    nflag=""
919 fi
920
921 if [ -n "$pflag" ]; then
922    pflag=$pflag
923 else
924    pflag=""
925 fi
926
927 if [ -n "$nflag" ]; then
928    nflag=$nflag
929 else
930    nflag=""
931 fi
932
933 if [ -n "$pflag" ]; then
934    pflag=$pflag
935 else
936    pflag=""
937 fi
938
939 if [ -n "$nflag" ]; then
940    nflag=$nflag
941 else
942    nflag=""
943 fi
944
945 if [ -n "$pflag" ]; then
946    pflag=$pflag
947 else
948    pflag=""
949 fi
950
951 if [ -n "$nflag" ]; then
952    nflag=$nflag
953 else
954    nflag=""
955 fi
956
957 if [ -n "$pflag" ]; then
958    pflag=$pflag
959 else
960    pflag=""
961 fi
962
963 if [ -n "$nflag" ]; then
964    nflag=$nflag
965 else
966    nflag=""
967 fi
968
969 if [ -n "$pflag" ]; then
970    pflag=$pflag
971 else
972    pflag=""
973 fi
974
975 if [ -n "$nflag" ]; then
976    nflag=$nflag
977 else
978    nflag=""
979 fi
980
981 if [ -n "$pflag" ]; then
982    pflag=$pflag
983 else
984    pflag=""
985 fi
986
987 if [ -n "$nflag" ]; then
988    nflag=$nflag
989 else
990    nflag=""
991 fi
992
993 if [ -n "$pflag" ]; then
994    pflag=$pflag
995 else
996    pflag=""
997 fi
998
999 if [ -n "$nflag" ]; then
1000    nflag=$nflag
1001 else
1002    nflag=""
1003 fi
1004
1005 if [ -n "$pflag" ]; then
1006    pflag=$pflag
1007 else
1008    pflag=""
1009 fi
1010
1011 if [ -n "$nflag" ]; then
1012    nflag=$nflag
1013 else
1014    nflag=""
1015 fi
1016
1017 if [ -n "$pflag" ]; then
1018    pflag=$pflag
1019 else
1020    pflag=""
1021 fi
1022
1023 if [ -n "$nflag" ]; then
1024    nflag=$nflag
1025 else
1026    nflag=""
1027 fi
1028
1029 if [ -n "$pflag" ]; then
1030    pflag=$pflag
1031 else
1032    pflag=""
1033 fi
1034
1035 if [ -n "$nflag" ]; then
1036    nflag=$nflag
1037 else
1038    nflag=""
1039 fi
1040
1041 if [ -n "$pflag" ]; then
1042    pflag=$pflag
1043 else
1044    pflag=""
1045 fi
1046
1047 if [ -n "$nflag" ]; then
1048    nflag=$nflag
1049 else
1050    nflag=""
1051 fi
1052
1053 if [ -n "$pflag" ]; then
1054    pflag=$pflag
1055 else
1056    pflag=""
1057 fi
1058
1059 if [ -n "$nflag" ]; then
1060    nflag=$nflag
1061 else
1062    nflag=""
1063 fi
1064
1065 if [ -n "$pflag" ]; then
1066    pflag=$pflag
1067 else
1068    pflag=""
1069 fi
1070
1071 if [ -n "$nflag" ]; then
1072    nflag=$nflag
1073 else
1074    nflag=""
1075 fi
1076
1077 if [ -n "$pflag" ]; then
1078    pflag=$pflag
1079 else
1080    pflag=""
1081 fi
1082
1083 if [ -n "$nflag" ]; then
1084    nflag=$nflag
1085 else
1086    nflag=""
1087 fi
1088
1089 if [ -n "$pflag" ]; then
1090    pflag=$pflag
1091 else
1092    pflag=""
1093 fi
1094
1095 if [ -n "$nflag" ]; then
1096    nflag=$nflag
1097 else
1098    nflag=""
1099 fi
1100
1101 if [ -n "$pflag" ]; then
1102    pflag=$pflag
1103 else
1104    pflag=""
1105 fi
1106
1107 if [ -n "$nflag" ]; then
1108    nflag=$nflag
1109 else
1110    nflag=""
1111 fi
1112
1113 if [ -n "$pflag" ]; then
1114    pflag=$pflag
1115 else
1116    pflag=""
1117 fi
1118
1119 if [ -n "$nflag" ]; then
1120    nflag=$nflag
1121 else
1122    nflag=""
1123 fi
1124
1125 if [ -n "$pflag" ]; then
1126    pflag=$pflag
1127 else
1128    pflag=""
1129 fi
1130
1131 if [ -n "$nflag" ]; then
1132    nflag=$nflag
1133 else
1134    nflag=""
1135 fi
1136
1137 if [ -n "$pflag" ]; then
1138    pflag=$pflag
1139 else
1140    pflag=""
1141 fi
1142
1143 if [ -n "$nflag" ]; then
1144    nflag=$nflag
1145 else
1146    nflag=""
1147 fi
1148
1149 if [ -n "$pflag" ]; then
1150    pflag=$pflag
1151 else
1152    pflag=""
1153 fi
1154
1155 if [ -n "$nflag" ]; then
1156    nflag=$nflag
1157 else
1158    nflag=""
1159 fi
1160
1161 if [ -n "$pflag" ]; then
1162    pflag=$pflag
1163 else
1164    pflag=""
1165 fi
1166
1167 if [ -n "$nflag" ]; then
1168    nflag=$nflag
1169 else
1170    nflag=""
1171 fi
1172
1173 if [ -n "$pflag" ]; then
1174    pflag=$pflag
1175 else
1176    pflag=""
1177 fi
1178
1179 if [ -n "$nflag" ]; then
1180    nflag=$nflag
1181 else
1182    nflag=""
1183 fi
1184
1185 if [ -n "$pflag" ]; then
1186    pflag=$pflag
1187 else
1188    pflag=""
1189 fi
1190
1191 if [ -n "$nflag" ]; then
1192    nflag=$nflag
1193 else
1194    nflag=""
1195 fi
1196
1197 if [ -n "$pflag" ]; then
1198    pflag=$pflag
1199 else
1200    pflag=""
1201 fi
1202
1203 if [ -n "$nflag" ]; then
1204    nflag=$nflag
1205 else
1206    nflag=""
1207 fi
1208
1209 if [ -n "$pflag" ]; then
1210    pflag=$pflag
1211 else
1212    pflag=""
1213 fi
1214
1215 if [ -n "$nflag" ]; then
1216    nflag=$nflag
1217 else
1218    nflag=""
1219 fi
1220
1221 if [ -n "$pflag" ]; then
1222    pflag=$pflag
1223 else
1224    pflag=""
1225 fi
1226
1227 if [ -n "$nflag" ]; then
1228    nflag=$nflag
1229 else
1230    nflag=""
1231 fi
1232
1233 if [ -n "$pflag" ]; then
1234    pflag=$pflag
1235 else
1236    pflag=""
1237 fi
1238
1239 if [ -n "$nflag" ]; then
1240    nflag=$nflag
1241 else
1242    nflag=""
1243 fi
1244
1245 if [ -n "$pflag" ]; then
1246    pflag=$pflag
1247 else
1248    pflag=""
1249 fi
1250
1251 if [ -n "$nflag" ]; then
1252    nflag=$nflag
1253 else
1254    nflag=""
1255 fi
1256
1257 if [ -n "$pflag" ]; then
1258    pflag=$pflag
1259 else
1260    pflag=""
1261 fi
1262
1263 if [ -n "$nflag" ]; then
1264    nflag=$nflag
1265 else
1266    nflag=""
1267 fi
1268
1269 if [ -n "$pflag" ]; then
1270    pflag=$pflag
1271 else
1272    pflag=""
1273 fi
1274
1275 if [ -n "$nflag" ]; then
1276    nflag=$nflag
1277 else
1278    nflag=""
1279 fi
1280
1281 if [ -n "$pflag" ]; then
1282    pflag=$pflag
1283 else
1284    pflag=""
1285 fi
1286
1287 if [ -n "$nflag" ]; then
1288    nflag=$nflag
1289 else
1290    nflag=""
1291 fi
1292
1293 if [ -n "$pflag" ]; then
1294    pflag=$pflag
1295 else
1296    pflag=""
1297 fi
1298
1299 if [ -n "$nflag" ]; then
1300    nflag=$nflag
1301 else
1302    nflag=""
1303 fi
1304
1305 if [ -n "$pflag" ]; then
1306    pflag=$pflag
1307 else
1308    pflag=""
1309 fi
1310
1311 if [ -n "$nflag" ]; then
1312    nflag=$nflag
1313 else
1314    nflag=""
1315 fi
1316
1317 if [ -n "$pflag" ]; then
1318    pflag=$pflag
1319 else
1320    pflag=""
1321 fi
1322
1323 if [ -n "$nflag" ]; then
1324    nflag=$nflag
1325 else
1326    nflag=""
1327 fi
1328
1329 if [ -n "$pflag" ]; then
1330    pflag=$pflag
1331 else
1332    pflag=""
1333 fi
1334
1335 if [ -n "$nflag" ]; then
1336    nflag=$nflag
1337 else
1338    nflag=""
1339 fi
1340
1341 if [ -n "$pflag" ]; then
1342    pflag=$pflag
1343 else
1344    pflag=""
1345 fi
1346
1347 if [ -n "$nflag" ]; then
1348    nflag=$nflag
1349 else
1350    nflag=""
1351 fi
1352
1353 if [ -n "$pflag" ]; then
1354    pflag=$pflag
1355 else
1356    pflag=""
1357 fi
1358
1359 if [ -n "$nflag" ]; then
1360    nflag=$nflag
1361 else
1362    nflag=""
1363 fi
1364
1365 if [ -n "$pflag" ]; then
1366    pflag=$pflag
1367 else
1368    pflag=""
1369 fi
1370
1371 if [ -n "$nflag" ]; then
1372    nflag=$nflag
1373 else
1374    nflag=""
1375 fi
1376
1377 if [ -n "$pflag" ]; then
1378    pflag=$pflag
1379 else
1380    pflag=""
1381 fi
1382
1383 if [ -n "$nflag" ]; then
1384    nflag=$nflag
1385 else
1386    nflag=""
1387 fi
1388
1389 if [ -n "$pflag" ]; then
1390    pflag=$pflag
1391 else
1392    pflag=""
1393 fi
1394
1395 if [ -n "$nflag" ]; then
1396    nflag=$nflag
1397 else
1398    nflag=""
1399 fi
1400
1401 if [ -n "$pflag" ]; then
1402    pflag=$pflag
1403 else
1404    pflag=""
1405 fi
1406
1407 if [ -n "$nflag" ]; then
1408    nflag=$nflag
1409 else
1410    nflag=""
1411 fi
1412
1413 if [ -n "$pflag" ]; then
1414    pflag=$pflag
1415 else
1416    pflag=""
1417 fi
1418
1419 if [ -n "$nflag" ]; then
1420    nflag=$nflag
1421 else
1422    nflag=""
1423 fi
1424
1425 if [ -n "$pflag" ]; then
1426    pflag=$pflag
1427 else
1428    pflag=""
1429 fi
1430
1431 if [ -n "$nflag" ]; then
1432    nflag=$nflag
1433 else
1434    nflag=""
1435 fi
1436
1437 if [ -n "$pflag" ]; then
1438    pflag=$pflag
1439 else
1440    pflag=""
1441 fi
1442
1443 if [ -n "$nflag" ]; then
1444    nflag=$nflag
1445 else
1446    nflag=""
1447 fi
1448
1449 if [ -n "$pflag" ]; then
1450    pflag=$pflag
1451 else
1452    pflag=""
1453 fi
1454
1455 if [ -n "$nflag" ]; then
1456    nflag=$nflag
1457 else
1458    nflag=""
1459 fi
1460
1461 if [ -n "$pflag" ]; then
1462    pflag=$pflag
1463 else
1464    pflag=""
1465 fi
1466
1467 if [ -n "$nflag" ]; then
1468    nflag=$nflag
1469 else
1470    nflag=""
1471 fi
1472
1473 if [ -n "$pflag" ]; then
1474    pflag=$pflag
1475 else
1476    pflag=""
1477 fi
1478
1479 if [ -n "$nflag" ]; then
1480    nflag=$nflag
1481 else
1482    nflag=""
1483 fi
1484
1485 if [ -n "$pflag" ]; then
1486    pflag=$pflag
1487 else
1488    pflag=""
1489 fi
1490
1491 if [ -n "$nflag" ]; then
1492    nflag=$nflag
1493 else
1494    nflag=""
1495 fi
1496
1497 if [ -n "$pflag" ]; then
1498    pflag=$pflag
1499 else
1500    pflag=""
1501 fi
1502
1503 if [ -n "$
```



```
mohamedlamine@fedora: ~/lab11
[mohamedlamine@fedora ~]$ cd lab11
[mohamedlamine@fedora lab11]$ bash prog11.1.sh -p ky -i input.txt -o output.txt
-c -n
grep: input.txt: Нет такого файла или каталога
[mohamedlamine@fedora lab11]$ touch input.txt
[mohamedlamine@fedora lab11]$ bash prog11.1.sh -p ky -i input.txt -o output.txt
-c -n
[mohamedlamine@fedora lab11]$ bash prog11.1.sh -p ky -i input.txt -o output.txt
-c -n
[mohamedlamine@fedora lab11]$
```

Рис. 4.2: Вызов программы в терминале

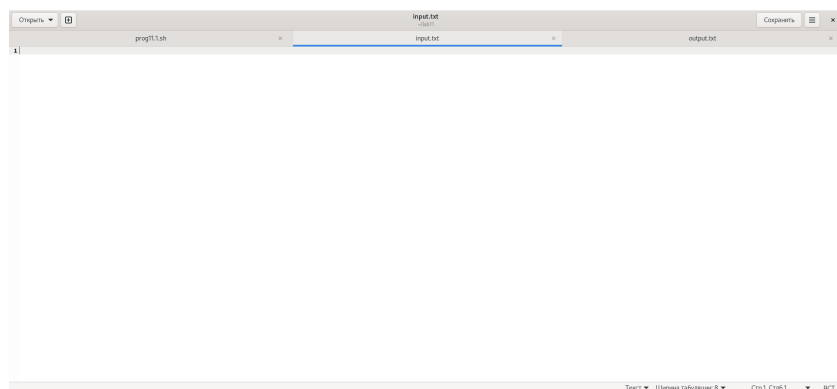


Рис. 4.3: Результат

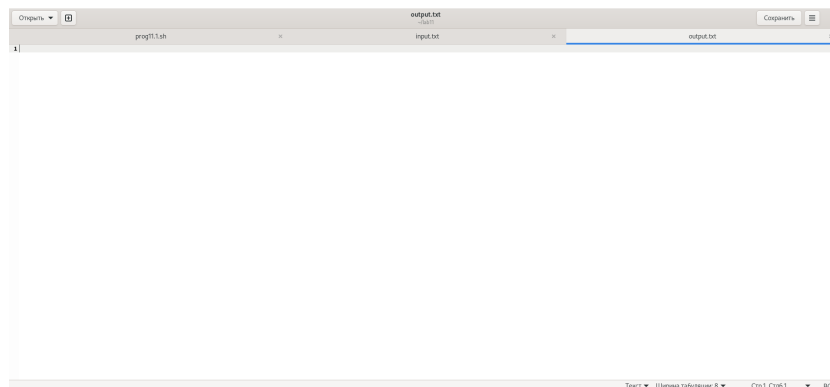


Рис. 4.4: Результат

2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено. (рис. [4.5])

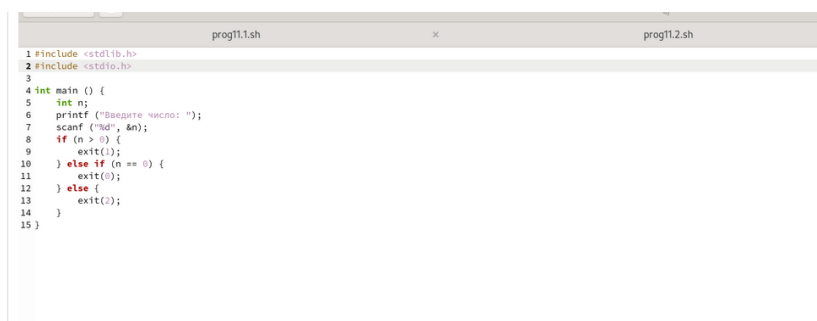


Рис. 4.5: Вторая программа

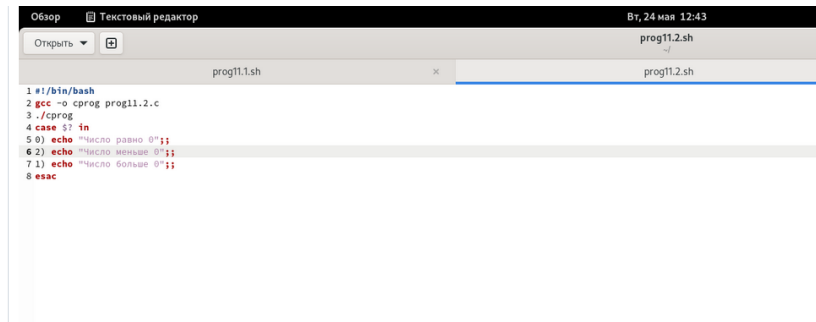


Рис. 4.6: Вторая программа

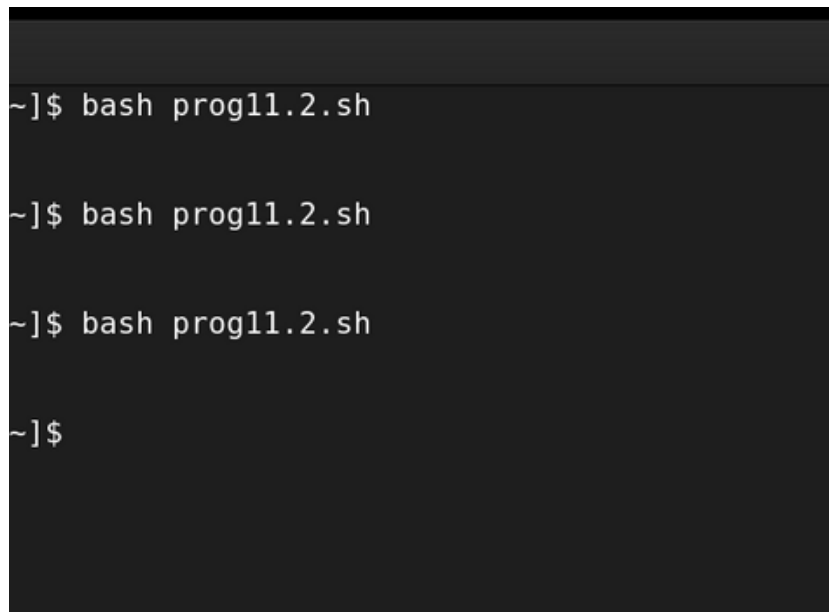


Рис. 4.7: Результат

3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют). (рис. [4.8])

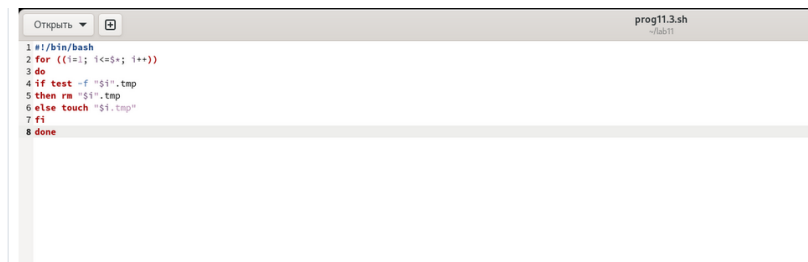


Рис. 4.8: Третья программа

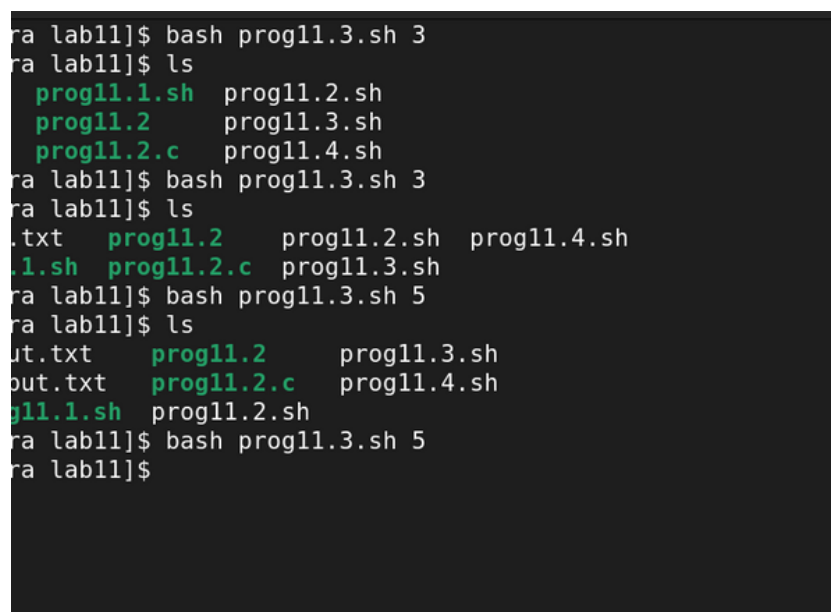


Рис. 4.9: Результат

4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). (рис. [4.10])



Рис. 4.10: Четвертая программа

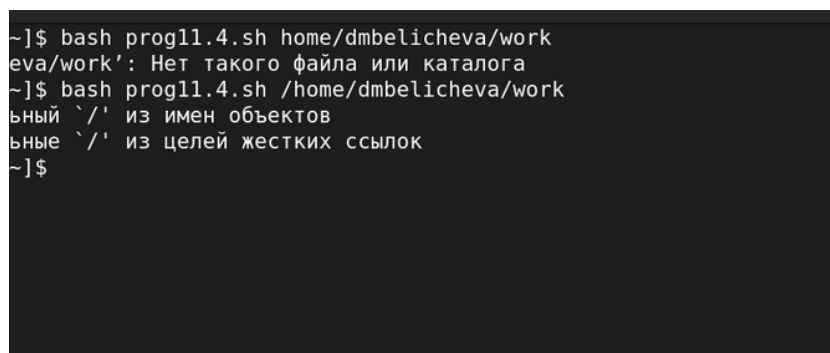


Рис. 4.11: Вызов программы в терминале

Результат

5 Выводы

В процессе выполнения данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

6 Контрольные вопросы

1. Каково предназначение команды `getopts`?

Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае:

```
while
getopts o:i:Ltr optletter do
case $optletter in
o) iflag = 1; oval =OPTARG;;
i) iflag=1; ival=$OPTARG;;
L) Lflag=1;;
t) tflag=1;;
r) rflag=1;;
*) echo Illegal option
$optletter
esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равно `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следова-

тельно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имён файлов текущего каталога можно использовать следующие символы: – соответствует произвольной, в том числе и пустой строке; ? – соответствует любому одинарному символу; [с1-с2] – соответствует любому символу, лексикографически находящемуся между символами с1 и с2. Например, `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .с` – *выведет все файлы с последними двумя символами, совпадающими с .с*. `echo prog.?` – *выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog..* `[a-z]` – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3. Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

4. Какие операторы используются для прерывания цикла?

Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

5. Для чего нужны команды `false` и `true`?

Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т.е. ложь).

6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?

Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

7. Объясните различия между конструкциями `while` и `until`.

Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда

из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.

Список литературы