

系统调用追踪

问题：

针对 <https://de4dcr0w.github.io/%E5%88%86%E6%9E%90fork%E7%B3%BB%E7%BB%9F%E8%B0%83%E7%94%A8.html>, fork 系统调用, linux-4.20 内核, glibc-2.23 版, 通过 GDB 调试跟踪, 给出跟踪过程, 并给出自己的心得体会。

搭建 linux-4.20 内核：

1. 下载 linux-4.20 源代码：linux kernel 的官方网站为 <https://kernel.org/>, 由于是海外网站, 下载速度比较慢, 我去国内镜像源找了找, 在阿里云镜像网站 <https://developer.aliyun.com/mirror/?serviceType=mirror> 的 linux-kernel 镜像找到了 linux-4.20 的备份 <https://mirrors.aliyun.com/linux-kernel/v4.x/linux-4.20.tar.gz>, 下载速度很快。

2. 解压, 我把他解压到了用户家目录, 命令为:

```
tar -zxvf ~/下载/ linux-4.20.tar.gz -C ~
```

3. 编译, 依照 https://mp.weixin.qq.com/s/STExqTiKpGXSp9E_DerV6g

遇到了问题

1. configure: error: Building GCC requires GMP 4.2+, MPFR 2.4.0+ and MPC 0.8.0+.

见 <http://www.cnblogs.com/gyfluck/p/10537436.html>。

2. [error: '-mindirect-branch' and '-fcf-protection' are not compatible]

原因可能是新版本 gcc 的一些更改出现了 bug, 重新安装一遍 gcc-8, 可以成功运行。

依照 <https://blog.csdn.net/linuxarm123/article/details/99292991> 来安装旧版本 gcc, 源代码可以去阿里云镜像 <http://mirrors.aliyun.com/gnu/gcc> 下寻找。

安装文件解压到家目录即可, 只需在 make install 时使用 sudo, 不必解压到 root 目录

在 configure 时建议加入参数 `--prefix=/usr/local/gcc-8.x` 来设置安装目录, 不然可能会发生覆盖

安装完成后可以去安装目录下的 bin 目录中使用旧版本 gcc, 默认为旧版本参见 <https://blog.csdn.net/u012822903/article/details/68934793> 的第 6 条, <https://blog.csdn.net/mo4776/article/details/119837501> 的“指定 gcc/g++, glibc 的版本进行编译”。

3. configure: error: cannot compute suffix of object files: cannot compile

见 https://blog.csdn.net/testcs_dn/article/details/45437149。

4. rror: dereferencing pointer to incomplete type 'struct ucontext'

见 https://blog.csdn.net/weixin_46584887/article/details/122527357。

5. fatal error: sys/ustat.h: No such file or directory

见 https://blog.csdn.net/weixin_46584887/article/details/122538399。

4. 检查，依照 https://mp.weixin.qq.com/s/STExqTiKpGXSp9E_DerV6g，其实文中的 rootfs.img 虚拟硬盘文件可以做得更大一点，以便我们向其中放入更多程序和文件，系统首次加载此虚拟硬盘时是只读的，可以重新挂载：`mount -o remount rw /` 解决。

```
/home # uname -r
4.20.0
/home #
```

成功。

安装 glibc-2.23:

1. 下载，阿里云的镜像：<https://mirrors.aliyun.com/gnu/glibc/glibc-2.23.tar.gz>。
2. 安装，见 <https://blog.csdn.net/u010835747/article/details/109624445>
3. 检查，见 <https://www.it1352.com/2686964.html>，这里我编译 glibc-version.c 时指定了 lib 的目录（glibc-2.23 安装目录下的 lib 目录，我这里已经在 glibc 目录下了，所以是 lib）并设置为了静态连接

```
gcc -L./lib -O0 -static ~/文档/glibc-version.c -o ~/文档/glibc-version_static
```

这样内核系统就不用再安装 glibc 了，程序可以直接在内核系统运行，只是程序会比较大。

```
monkey@monkey-JinGang-T2-S:~/shadisk/share/glibc-2.23$ gcc -L./lib -O0 -static ~/文档/glibc-version.
monkey@monkey-JinGang-T2-S:~/shadisk/share/glibc-2.23$ ./文档/glibc-version_static
GNU libc version: 2.23
monkey@monkey-JinGang-T2-S:~/shadisk/share/glibc-2.23$
```

成功。

fork 测试文件

1. 编译测试程序：根据题目提供的网址 <https://de4dcr0w.github.io/%E5%88%86%E6%9E%90fork%E7%B3%BB%E7%BB%9F%E8%B0%83%E7%94%A8.html> 给出的代码，我进行了拷贝编译，编译参数和 glibc-version 一样

```
gcc -L./lib -O0 -static ~/文档/fork.c -o ~/文档/fork_static
gcc -L./lib -O0 -static ~/文档/fork-asm.c -o ~/文档/fork-asm_static
```

2. 检查运行状况：将编译好的文件放入内核系统的虚拟硬盘（rootfs.img）中，启动内核系统并运行，运行结果如下：

```

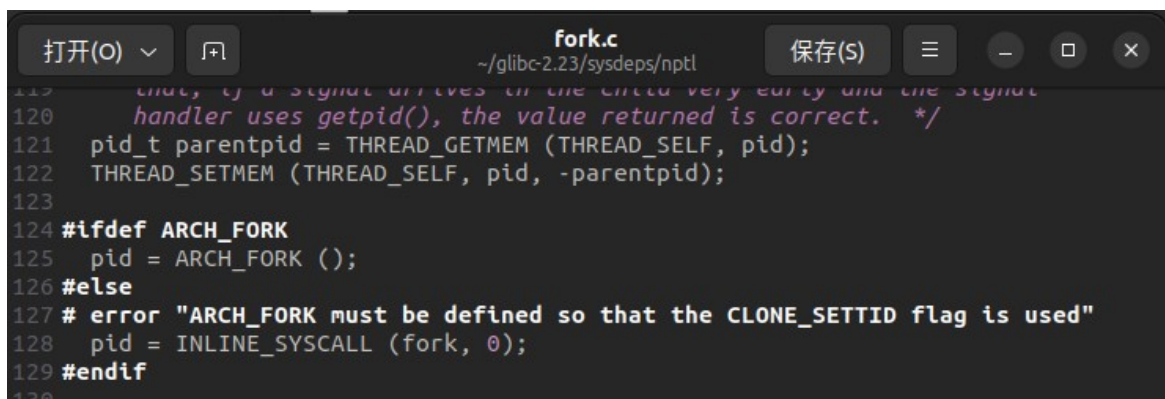
/home # ./fork_static
it's a parent process,my process id is 1092
count:1
it's a child process,my process id is 1093
count:1
/home # ./fork-asm_static
it's a parent process,my process id is 1094
count:1
it's a child process,my process id is 1095
count:1
/home # 

```

成功。

解析 fork() 源代码

1. 通过搜索 glibc 目录下的 fork.c，找到了 fork 的函数定义，实际为__fork__函数：

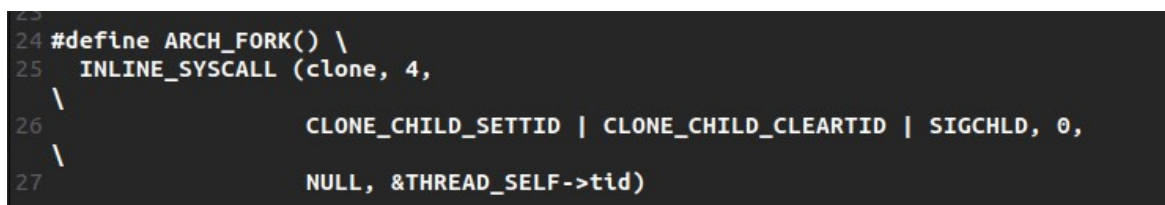


```

fork.c
~/glibc-2.23/sysdeps/nptl
保存(S)
119  /* If a signal arrives on the child very early and the signal
120     handler uses getpid(), the value returned is correct. */
121  pid_t parentpid = THREAD_GETMEM (THREAD_SELF, pid);
122  THREAD_SETMEM (THREAD_SELF, pid, -parentpid);
123
124  #ifdef ARCH_FORK
125    pid = ARCH_FORK ();
126  #else
127    # error "ARCH_FORK must be defined so that the CLONE_SETTID flag is used"
128    pid = INLINE_SYSCALL (fork, 0);
129  #endif
130

```

发现 fork 调用了 ARCH_FORK() 位于 ~/glibc-2.23/sysdeps/unix/sysv/linux/x86_64/arch-fork.h



```

23
24 #define ARCH_FORK() \
25   INLINE_SYSCALL (clone, 4,
26   \
27   \
28   \
29   \
30   \
31   \
32   \
33   \
34   \
35   \
36   \
37   \
38   \
39   \
40   \
41   \
42   \
43   \
44   \
45   \
46   \
47   \
48   \
49   \
50   \
51   \
52   \
53   \
54   \
55   \
56   \
57   \
58   \
59   \
60   \
61   \
62   \
63   \
64   \
65   \
66   \
67   \
68   \
69   \
70   \
71   \
72   \
73   \
74   \
75   \
76   \
77   \
78   \
79   \
80   \
81   \
82   \
83   \
84   \
85   \
86   \
87   \
88   \
89   \
90   \
91   \
92   \
93   \
94   \
95   \
96   \
97   \
98   \
99   \
100  \
101  \
102  \
103  \
104  \
105  \
106  \
107  \
108  \
109  \
110  \
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 \
1001 \
1002 \
1003 \
1004 \
1005 \
1006 \
1007 \
1008 \
1009 \
1010 \
1011 \
1012 \
1013 \
1014 \
1015 \
1016 \
1017 \
1018 \
1019 \
1020 \
1021 \
1022 \
1023 \
1024 \
1025 \
1026 \
1027 \
1028 \
1029 \
1030 \
1031 \
1032 \
1033 \
1034 \
1035 \
1036 \
1037 \
1038 \
1039 \
1040 \
1041 \
1042 \
1043 \
1044 \
1045 \
1046 \
1047 \
1048 \
1049 \
1050 \
1051 \
1052 \
1053 \
1054 \
1055 \
1056 \
1057 \
1058 \
1059 \
1060 \
1061 \
1062 \
1063 \
1064 \
1065 \
1066 \
1067 \
1068 \
1069 \
1070 \
1071 \
1072 \
1073 \
1074 \
1075 \
1076 \
1077 \
1078 \
1079 \
1080 \
1081 \
1082 \
1083 \
1084 \
1085 \
1086 \
1087 \
1088 \
1089 \
1090 \
1091 \
1092 \
1093 \
1094 \
1095 \
1096 \
1097 \
1098 \
1099 \
1100 \
1101 \
1102 \
1103 \
1104 \
1105 \
1106 \
1107 \
1108 \
1109 \
1110 \
1111 \
1112 \
1113 \
1114 \
1115 \
1116 \
1117 \
1118 \
1119 \
1120 \
1121 \
1122 \
1123 \
1124 \
1125 \
1126 \
1127 \
1128 \
1129 \
1130 \
1131 \
1132 \
1133 \
1134 \
1135 \
1136 \
1137 \
1138 \
1139 \
1140 \
1141 \
1142 \
1143 \
1144 \
1145 \
1146 \
1147 \
1148 \
1149 \
1150 \
1151 \
1152 \
1153 \
1154 \
1155 \
1156 \
1157 \
1158 \
1159 \
1160 \
1161 \
1162 \
1163 \
1164 \
1165 \
1166 \
1167 \
1168 \
1169 \
1170 \
1171 \
1172 \
1173 \
1174 \
1175 \
1176 \
1177 \
1178 \
1179 \
1180 \
1181 \
1182 \
1183 \
1184 \
1185 \
1186 \
1187 \
1188 \
1189 \
1190 \
1191 \
1192 \
1193 \
1194 \
1195 \
1196 \
1197 \
1198 \
1199 \
1200 \
1201 \
1202 \
1203 \
1204 \
1205 \
1206 \
1207 \
1208 \
1209 \
1210 \
1211 \
1212 \
1213 \
1214 \
1215 \
1216 \
1217 \
1218 \
1219 \
1220 \
1221 \
1222 \
1223 \
1224 \
1225 \
1226 \
1227 \
1228 \
1229 \
1230 \
1231 \
1232 \
1233 \
1234 \
1235 \
1236 \
1237 \
1238 \
1239 \
1240 \
1241 \
1242 \
1243 \
1244 \
1245 \
1246 \
1247 \
1248 \
1249 \
1250 \
1251 \
1252 \
1253 \
1254 \
1255 \
1256 \
1257 \
1258 \
1259 \
1260 \
1261 \
1262 \
1263 \
1264 \
1265 \
1266 \
1267 \
1268 \
1269 \
1270 \
1271 \
1272 \
1273 \
1274 \
1275 \
1276 \
1277 \
1278 \
1279 \
1280 \
1281 \
1282 \
1283 \
1284 \
1285 \
1286 \
1287 \
1288 \
1289 \
1290 \
1291 \
1292 \
1293 \
1294 \
1295 \
1296 \
1297 \
1298 \
1299 \
1300 \
1301 \
1302 \
1303 \
1304 \
1305 \
1306 \
1307 \
1308 \
1309 \
1310 \
1311 \
1312 \
1313 \
1314 \
1315 \
1316 \
1317 \
1318 \
1319 \
1320 \
1321 \
1322 \
1323 \
1324 \
1325 \
1326 \
1327 \
1328 \
1329 \
1330 \
1331 \
1332 \
1333 \
1334 \
1335 \
1336 \
1337 \
1338 \
1339 \
1340 \
1341 \
1342 \
1343 \
1344 \
1345 \
1346 \
1347 \
1348 \
1349 \
1350 \
1351 \
1352 \
1353 \
1354 \
1355 \
1356 \
1357 \
1358 \
1359 \
1360 \
1361 \
1362 \
1363 \
1364 \
1365 \
1366 \
1367 \
1368 \
1369 \
1370 \
1371 \
1372 \
1373 \
1374 \
1375 \
1376 \
1377 \
1378 \
1379 \
1380 \
1381 \
1382 \
1383 \
1384 \
1385 \
1386 \
1387 \
1388 \
1389 \
1390 \
1391 \
1392 \
1393 \
1394 \
1395 \
1396 \
1397 \
1398 \
1399 \
1400 \
1401 \
1402 \
1403 \
1404 \
1405 \
1406 \
1407 \
1408 \
1409 \
1410 \
1411 \
1412 \
1413 \
1414 \
1415 \
1416 \
1417 \
1418 \
1419 \
1420 \
1421 \
1422 \
1423 \
1424 \
1425 \
1426 \
1427 \
1428 \
1429 \
1430 \
1431 \
1432 \
1433 \
1434 \
1435 \
1436 \
1437 \
1438 \
1439 \
1440 \
1441 \
1442 \
1443 \
1444 \
1445 \
1446 \
1447 \
1448 \
1449 \
1450 \
1451 \
1452 \
1453 \
1454 \
1455 \
1456 \
1457 \
1458 \
1459 \
1460 \
1461 \
1462 \
1463 \
1464 \
1465 \
1466 \
1467 \
1468 \
1469 \
1470 \
1471 \
1472 \
1473 \
1474 \
1475 \
1476 \
1477 \
1478 \
1479 \
1480 \
1481 \
1482 \
1483 \
1484 \
1485 \
1486 \
1487 \
1488 \
1489 \
1490 \
1491 \
1492 \
1493 \
1494 \
1495 \
1496 \
1497 \
1498 \
1499 \
1500 \
1501 \
1502 \
1503 \
1504 \
1505 \
1506 \
1507 \
1508 \
1509 \
1510 \
1511 \
1512 \
1513 \
1514 \
1515 \
1516 \
1517 \
1518 \
1519 \
1520 \
1521 \
1522 \
1523 \
1524 \
1525 \
1526 \
1527 \
1528 \
1529 \
1530 \
1531 \
1532 \
1533 \
1534 \
1535 \
1536 \
1537 \
1538 \
1539 \
1540 \
1541 \
1542 \
1543 \
1544 \
1545 \
1546 \
1547 \
1548 \
1549 \
1550 \
1551 \
1552 \
1553 \
1554 \
1555 \
1556 \
1557 \
1558 \
1559 \
1560 \
1561 \
1562 \
1563 \
1564 \
1565 \
1566 \
1567 \
1568 \
1569 \
1570 \
1571 \
1572 \
1573 \
1574 \
1575 \
1576 \
1577 \
1578 \
1579 \
1580 \
1581 \
1582 \
1583 \
1584 \
1585 \
1586 \
1587 \
1588 \
1589 \
1590 \
1591 \
1592 \
1593 \
1594 \
1595 \
1596 \
1597 \
1598 \
1599 \
1600 \
1601 \
1602 \
1603 \
1604 \
1605 \
1606 \
1607 \
1608 \
1609 \
1610 \
1611 \
1612 \
1613 \
1614 \
1615 \
1616 \
1617 \
1618 \
1619 \
1620 \
1621 \
1622 \
1623 \
1624 \
1625 \
1626 \
1627 \
1628 \
1629 \
1630 \
1631 \
1632 \
1633 \
1634 \
1635 \
1636 \
1637 \
1638 \
1639 \
1640 \
1641 \
1642 \
1643 \
1644 \
1645 \
1646 \
1647 \
1648 \
1649 \
1650 \
1651 \
1652 \
1653 \
1654 \
1655 \
1656 \
1657 \
1658 \
1659 \
1660 \
1661 \
1662 \
1663 \
1664 \
1665 \
1666 \
1667 \
1668 \
1669 \
1670 \
1671 \
1672 \
1673 \
1674 \
1675 \
1676 \
1677 \
1678 \
1679 \
1680 \
1681 \
1682 \
1683 \
1684 \
1685 \
1686 \
1687 \
1688 \
1689 \
1690 \
1691 \
1692 \
1693 \
1694 \
1695 \
1696 \
1697 \
1698 \
1699 \
1700 \
1701 \
1702 \
1703 \
1704 \
1705 \
1706 \
1707 \
1708 \
1709 \
1710 \
1711 \
1712 \
1713 \
1714 \
1715 \
1716 \
1717 \
1718 \
1719 \
1720 \
1721 \
1722 \
1723 \
1724 \
1725 \
1726 \
1727 \
1728 \
1729 \
1730 \
1731 \
1732 \
1733 \
1734 \
1735 \
1736 \
1737 \
1738 \
1739 \
1740 \
1741 \
1742 \
1743 \
1744 \
1745 \
1746 \
1747 \
1748 \
1749 \
1750 \
1751 \
1752 \
1753 \
1754 \
1755 \
1756 \
1757 \
1758 \
1759 \
1760 \
1761 \
1762 \
1763 \
1764 \
1765 \
1766 \
1767 \
1768 \
1769 \
1770 \
1771 \
1772 \
1773 \
1774 \
1775 \
1776 \
1777 \
1778 \
1779 \
1780 \
1781 \
1782 \
1783 \
1784 \
1785 \
1786 \
1787 \
1788 \
1789 \
1790 \
1791 \
1792 \
1793 \
1794 \
1795 \
1796 \
1797 \
1798 \
1799 \
1800 \
1801 \
1802 \
1803 \
1804 \
1805 \
1806 \
1807 \
1808 \
1809 \
1810 \
1811 \
1812 \
1813 \
1814 \
1815 \
1816 \
1817 \
1818 \
1819 \
1820 \
1821 \
1822 \
1823 \
1824 \
1825 \
1826 \
1827 \
1828 \
1829 \
1830 \
1831 \
1832 \
1833 \
1834 \
1835 \
1836 \
1837 \
1838 \
1839 \
1840 \
1841 \
1842 \
1843 \
1844 \
1845 \
1846 \
1847 \
1848 \
1849 \
1850 \
1851 \
1852 \
1853 \
1854 \
1855 \
1856 \
1857 \
1858 \
1859 \
1860 \
1861 \
1862 \
1863 \
1864 \
1865 \
1866 \
1867 \
1868 \
1869 \
1870 \
1871 \
1872 \
1873 \
1874 \
1875 \
1876 \
1877 \
1878 \
1879 \
1880 \
1881 \
1882 \
1883 \
1884 \
1885 \
1886 \
1887 \
1888 \
1889 \
1890 \
1891 \
1892 \
1893 \
1894 \
1895 \
1896 \
1897 \
1898 \
1899 \
1900 \
1901 \
1902 \
1903 \
1904 \
1905 \
1906 \
1907 \
1908 \
1909 \
1910 \
1911 \
1912 \
1913 \
1914 \
1915 \
1916 \
1917 \
1918 \
1919 \
1920 \
1921 \
1922 \
1923 \
1924 \
1925 \
1926 \
1927 \
1928 \
1929 \
1930 \
1931 \
1932 \
1933 \
1934 \
1935 \
1936 \
1937 \
1938 \
1939 \
1940 \
1941 \
1942 \
1943 \
1944 \
1945 \
1946 \
1947 \
1948 \
1949 \
1950 \
1951 \
1952 \
1953 \
1954 \
1955 \
1956 \
1957 \
1958 \
1959 \
1960 \
1961 \
1962 \
1963 \
1964 \
1965 \
1966 \
1967 \
1968 \
1969 \
1970 \
1971 \
1972 \
1973 \
1974 \
1975 \
197
```

```
sysdep.h
~/glibc-2.23/sysdeps/unix/sysv/linux/x86_64
保存(S)

190 #else /* !__ASSEMBLER__ */
191 /* Define a macro which expands inline into the wrapper code for a system
192    call. */
193 # undef INLINE_SYSCALL
194 # define INLINE_SYSCALL(name, nr, args...) \
195    ({
196        unsigned long int resultvar = INTERNAL_SYSCALL (name, , nr, args);
197        if (__glibc_unlikely (INTERNAL_SYSCALL_ERROR_P (resultvar, )))
198            {
199                /* ... */
200            }
201    })
```

```
sysdep.h
~/glibc-2.23/sysdeps/unix/sysv/linux/x86_64
保存(S)

233    (long int) resultvar; })
234 # undef INTERNAL_SYSCALL
235 # define INTERNAL_SYSCALL(name, err, nr, args...) \
236    INTERNAL_SYSCALL_NCS (__NR_##name, err, nr, ##args)
237
```

```
sysdep.h
~/glibc-2.23/sysdeps/unix/sysv/linux/x86_64
保存(S)

223
224 # define INTERNAL_SYSCALL_NCS(name, err, nr, args...) \
225    ({
226        unsigned long int resultvar;
227        LOAD_ARGS_##nr (args)
228        LOAD_REGS_##nr
229        asm volatile (
230            "syscall\n\t"
231            : "=a" (resultvar)
232            : "0" (name) ASM_ARGS_##nr : "memory", REGISTERS_CLOBBERED_BY_SYSCALL);
233        (long int) resultvar; })
234 # undef INTERNAL_SYSCALL
```

由于 arch-fork 实际上就是一个宏定义，并不是一个函数，所以不能使用 return 语句，因此 arch-fork 的返回值是通过括号内的语句直接返回的。通过一个判断，对其结果 resultvar 进行检查，若结果存在问题，则设置错误码，同时将结果置为-1。

“__NR_##name”，其中“##”是将##左右两边的标签组合在一起（token pasting or token concatenation）。被展开成为“__NR_clone”，此处“__NR_clone”也是一个宏，表示 clone 的系统调用号，通过“INTERNAL_SYSCALL_NCS”函数的第一个参数就把函数名与系统调用号联系了起来。关于系统调用号的定义可在“~/linux-4.20/arch/sh/include/uapi/asm/unistd_64.h”中查找，得到__NR_clone 为 120。

详见 <https://blog.csdn.net/u012927281/article/details/51540447>。

到此为止，系统通过 syscall 进入了系统调用，下面进入内核进行追踪。

使用 GDB 进行跟踪

1. 根据 linux 源代码 kernel/fork.c, 发现了系统调用 _do_fork, 以此为起始断点依次设定了若干可以跟踪 fork 系统调用的断点:

```
(gdb) info b
Num      Type      Disp Enb Address              What
1        breakpoint keep  y   0xffffffff8105fdd0 in _do_fork at kernel/fork.c:2196
2        breakpoint keep  y   <MULTIPLE>
2.1      breakpoint keep  y   0xffffffff8105dff0 in copy_process at kernel/fork.c:1677
2.2      breakpoint keep  y   0xffffffff8105fd2b in copy_process at ./include/linux/topology.h:80
2.3      breakpoint keep  y   0xffffffff8105fe44 in copy_process at kernel/fork.c:1695
3        breakpoint keep  y   0xffffffff8105e134 in dup_task_struct at kernel/fork.c:846
4        breakpoint keep  y   0xffffffff81019b00 in copy_thread_tls
                    at ./arch/x86/include/asm/current.h:15
5        breakpoint keep  y   0xffffffff81c001d0 arch/x86/entry/entry_64.S:335
```

2. 跟踪 fork.c

跟踪到了到了两个相关的系统调用栈

```
I.
#0 copy_thread_tls (clone_flags=18874385, sp=0, arg=0, p=0xffff888006bd3e80, tls=0)
   at ./arch/x86/include/asm/current.h:15
#1 0xffffffff8105eccd in copy_process (clone_flags=18874385,
   stack_start=<optimized out>, stack_size=<optimized out>,
   child_tidptr=<optimized out>, pid=<optimized out>, trace=<optimized out>,
   tls=<optimized out>, node=<optimized out>) at kernel/fork.c:1931
#2 0xffffffff8105fead in copy_process (pid=<optimized out>, node=<optimized out>,
   tls=<optimized out>, trace=<optimized out>, child_tidptr=<optimized out>,
   stack_size=<optimized out>, stack_start=<optimized out>,
   clone_flags=<optimized out>) at kernel/fork.c:1713
#3 _do_fork (clone_flags=18874385, stack_start=<optimized out>,
   stack_size=<optimized out>, parent_tidptr=0x0 <irq_stack_union>,
   child_tidptr=<optimized out>, tls=<optimized out>) at kernel/fork.c:2221
#4 0xffffffff810024f3 in do_syscall_64 (nr=<optimized out>,
   regs=0x0 <irq_stack_union>) at arch/x86/entry/common.c:290
#5 0xffffffff81c0007c in entry_SYSCALL_64 () at arch/x86/entry/entry_64.S:175
#6 0x0000000000000000 in ?? ()

II.
#0 copy_thread_tls (clone_flags=18874385, sp=0, arg=0, p=0xffff888006bd4b00, tls=0)
   at ./arch/x86/include/asm/current.h:15
#1 0xffffffff8105eccd in copy_process (clone_flags=18874385,
   stack_start=<optimized out>, stack_size=<optimized out>,
   child_tidptr=<optimized out>, pid=<optimized out>, trace=<optimized out>,
   tls=<optimized out>, node=<optimized out>) at kernel/fork.c:1931
#2 0xffffffff8105fead in copy_process (pid=<optimized out>, node=<optimized out>,
   tls=<optimized out>, trace=<optimized out>, child_tidptr=<optimized out>,
   stack_size=<optimized out>, stack_start=<optimized out>,
   clone_flags=<optimized out>) at kernel/fork.c:1713
#3 _do_fork (clone_flags=18874385, stack_start=<optimized out>,
   stack_size=<optimized out>, parent_tidptr=0x0 <irq_stack_union>,
   child_tidptr=<optimized out>, tls=<optimized out>) at kernel/fork.c:2221
#4 0xffffffff810024f3 in do_syscall_64 (nr=<optimized out>,
   regs=0x0 <irq_stack_union>) at arch/x86/entry/common.c:290
#5 0xffffffff81c0007c in entry_SYSCALL_64 () at arch/x86/entry/entry_64.S:175
#6 0x0000000000000000 in ?? ()
```


通过和执行 ls 跟踪到的 fork 调用栈对比, 可知, 我们得到的 (I.) 调用栈是 shell 执行外部命令时调用的, shell 解释器会调用 fork 自身的一个拷贝, 然后调用 exec 系列函数来执行外部命令, 这样外部命令就取代了先前的子 shell。

我们只需观察执行命令后得到的第二个系统调用栈即可。

```
Breakpoint 1, _do_fork (clone_flags=18874385, stack_start=0, stack_size=0, parent_tidptr=0x0 <irq_stack_union>, child_tidptr=0x8f5b50, tls=0) at kernel/fork.c:2196
2196      {
```

停在 do_fork 处。可以看到 do_fork 有很多参数, 其意义分别如下:

- clone_flags: 子进程创建相关标志, 通过此参数来对父进程的资源进行选择复制
- stack_start: 子进程用户堆栈的地址
- stack_size: 用户态堆栈的大小, 通常为 0
- parent_tidptr 和 child_tidpre: 父进程和子进程的 pid

```
Breakpoint 2, copy_process (pid=<optimized out>, node=<optimized out>, tls=<optimized out>, trace=<optimized out>, child_tidptr=<optimized out>, stack_size=<optimized out>, stack_start=<optimized out>, clone_flags=<optimized out>) at kernel/fork.c:1695
1695      if ((clone_flags & (CLONE_NEWNS|CLONE_FS)) == (CLONE_NEWNS|CLONE_FS))
(gdb)
Continuing.

Breakpoint 2, copy_process (clone_flags=18874385, stack_start=0, stack_size=0, child_tidptr=0x8f5b50, pid=0x0 <irq_stack_union>, trace=0, tls=0, node=-1) at kernel/fork.c:1677
1677      static __latent_entropy struct task_struct *copy_process(
```

copy_process 复制父进程信息, 获得 pid、调用 wake_up_new_task 将子进程加入调度队列等待分配 CPU 资源。

```
Breakpoint 3, dup_task_struct (node=-1, orig=<optimized out>) at kernel/fork.c:846
846      if (node == NUMA_NO_NODE)
```

copy_process 调用 dup_task_struct 复制父进程的数据结构、以及进行初始化、信息检查等工作;

```
Breakpoint 4, copy_thread_tls (clone_flags=18874385, sp=0, arg=0, p=0xfffff888006bd4b00, tls=0) at ./arch/x86/include/asm/current.h:15
```

copy_process 调用 copy_thread_tls 来初始化子进程的内核栈, 设置 pid。

```
Breakpoint 5, ret_from_fork () at arch/x86/entry/entry_64.S:335
335      movq    %rax, %rdi
```

最后从 ret_from_fork 开始执行, 直到结束。

3. 跟踪 fork-asm.c

```

#0  copy_thread_tls (clone_flags=17, sp=0, arg=0, p=0xfffff888006bd4b00, tls=0)
    at ./arch/x86/include/asm/current.h:15
#1  0xfffffffff8105eccd in copy_process (clone_flags=17, stack_start=<optimized out>,
    stack_size=<optimized out>, child_tidptr=<optimized out>, pid=<optimized out>,
    trace=<optimized out>, tls=<optimized out>, node=<optimized out>)
    at kernel/fork.c:1931
#2  0xfffffffff8105fead in copy_process (pid=<optimized out>, node=<optimized out>,
    tls=<optimized out>, trace=<optimized out>, child_tidptr=<optimized out>,
    stack_size=<optimized out>, stack_start=<optimized out>,
    clone_flags=<optimized out>) at kernel/fork.c:1713
#3  _do_fork (clone_flags=17, stack_start=<optimized out>, stack_size=<optimized out>,
    parent_tidptr=0x0 <irq_stack_union>, child_tidptr=<optimized out>,
    tls=<optimized out>) at kernel/fork.c:2221
#4  0xfffffffff810025eb in do_syscall_32_irqs_on (regs=<optimized out>)
    at arch/x86/entry/common.c:326
#5  do_int80_syscall_32 (regs=0x11 <irq_stack_union+17>) at arch/x86/entry/common.c:349
#6  0xfffffffff81c01500 in entry_INT80_compat () at arch/x86/entry/entry_64_compat.S:411
#7  0x0000000000000000 in ?? ()

```

查看 fork 中断的编号:

```

Breakpoint 6, do_int80_syscall_32 (regs=0xffffc900000eff58) at ./arch/x86/include/asm/irqflags.h:52
52      asm volatile("sti": : : "memory");
(gdb) info registers
rax                0x2                2

```

调用的参数:

```

Breakpoint 7, do_syscall_32_irqs_on (regs=<optimized out>) at arch/x86/entry/common.c:306
306      struct thread_info *ti = current_thread_info();
Breakpoint 1, _do_fork (clone_flags=17, stack_start=0, stack_size=0, parent_tidptr=0x0 <irq_stack_union>,
    child_tidptr=0x0 <irq_stack_union>, tls=0) at kernel/fork.c:2196
2196  {
Breakpoint 2, copy_process (pid=<optimized out>, node=<optimized out>, tls=<optimized out>, trace=<optimized out>,
    child_tidptr=<optimized out>, stack_size=<optimized out>, stack_start=<optimized out>, clone_flags=<optimized out>)
    at kernel/fork.c:1695
1695      if ((clone_flags & (CLONE_NEWNS|CLONE_FS)) == (CLONE_NEWNS|CLONE_FS))
Breakpoint 2, copy_process (clone_flags=17, stack_start=0, stack_size=0, child_tidptr=0x0 <irq_stack_union>,
    pid=0x0 <irq_stack_union>, trace=0, tls=0, node=-1) at kernel/fork.c:1677
1677  static __latent_entropy struct task_struct *copy_process(
Breakpoint 3, dup_task_struct (node=-1, orig=<optimized out>) at kernel/fork.c:846
846      if (node == NUMA_NO_NODE)
Breakpoint 4, copy_thread_tls (clone_flags=17, sp=0, arg=0, p=0xfffff888006bd4b00, tls=0)
    at ./arch/x86/include/asm/current.h:15
15      return this_cpu_read_stable(current_task);
Breakpoint 5, ret_from_fork () at arch/x86/entry/entry_64.S:335
335      movq    %rax, %rdi

```


系统作了优化，使用 int 80h 系统依旧通过 syscall 来进行系统调用的处理。

感悟

通过查阅源代码我体会到了 c 语言的高深莫测；系统调用的所跟踪出的内核函数复杂而且规模庞大，要读懂他们之间的联系必须很有耐心，逻辑清晰。