Andrea Abed

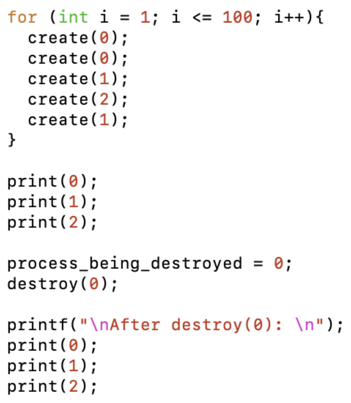
CS 480 - Professor Ramamoorthy

7 June 2024

Lab 1 Report - Process Creation and Destruction

I was able to successfully complete both versions of the lab, create the makefile, and pass all the tests on the check script made by Dat over 10 times. Watching the lectures on chapter 2, in addition to myself taking extensive notes about processes, process control blocks, the ways in which they can be stored, etc. from Zybooks, strengthened my knowledge about this topic and made me confident when going about this lab. Originally when I was starting out the lab, I was confused about when we would need to use dynamic memory allocation. This was because, from my learning from the Zybooks, I understood that an OS can organize all its PCBs one of two ways: with an array of structures or an array of pointers to dynamically allocated PCBs. Since this lab required us to make an array of structures, I did not think we needed to dynamically allocate memory for anything. Later I figured out that with the C programming language, we needed to dynamically allocate the memory when creating new child structures for the linked list of children, since the size of the list is not already known. A second problem I was encountering was that the first create(p) that I was calling, so create(0) would cause 0 to be a child of 0. After emailing and rereading the instructions, I realized that I needed to move forward with assuming process 0 is always an existing process and always takes the space of the first PCB structure in the array. Lastly, another big problem that I came across was regarding my destroy implementation. When I called destroy(p) and recursively called destroy on all of p’s descendants, it would end up not only terminating p’s descendants, but it would also terminate p itself which was not the intention. To solve this problem, I created a global int variable called “process_being_destroyed” which would be initialized with a value in the main method before calling destroy. This variable remembers the p that initially called destroy so that it only sets its children to null (for version 1) or its first child to -1 (for version 2) without emptying/terminating the entire process itself as it would do for all of its descendants. The sources that I used for this lab were the lectures, Zybooks, the helper video, Dat’s “Cs420's vscode tutorial” video, and mainly ww3 schools website for reminders about pointers and structures in C.

Both versions ran the following in main…..



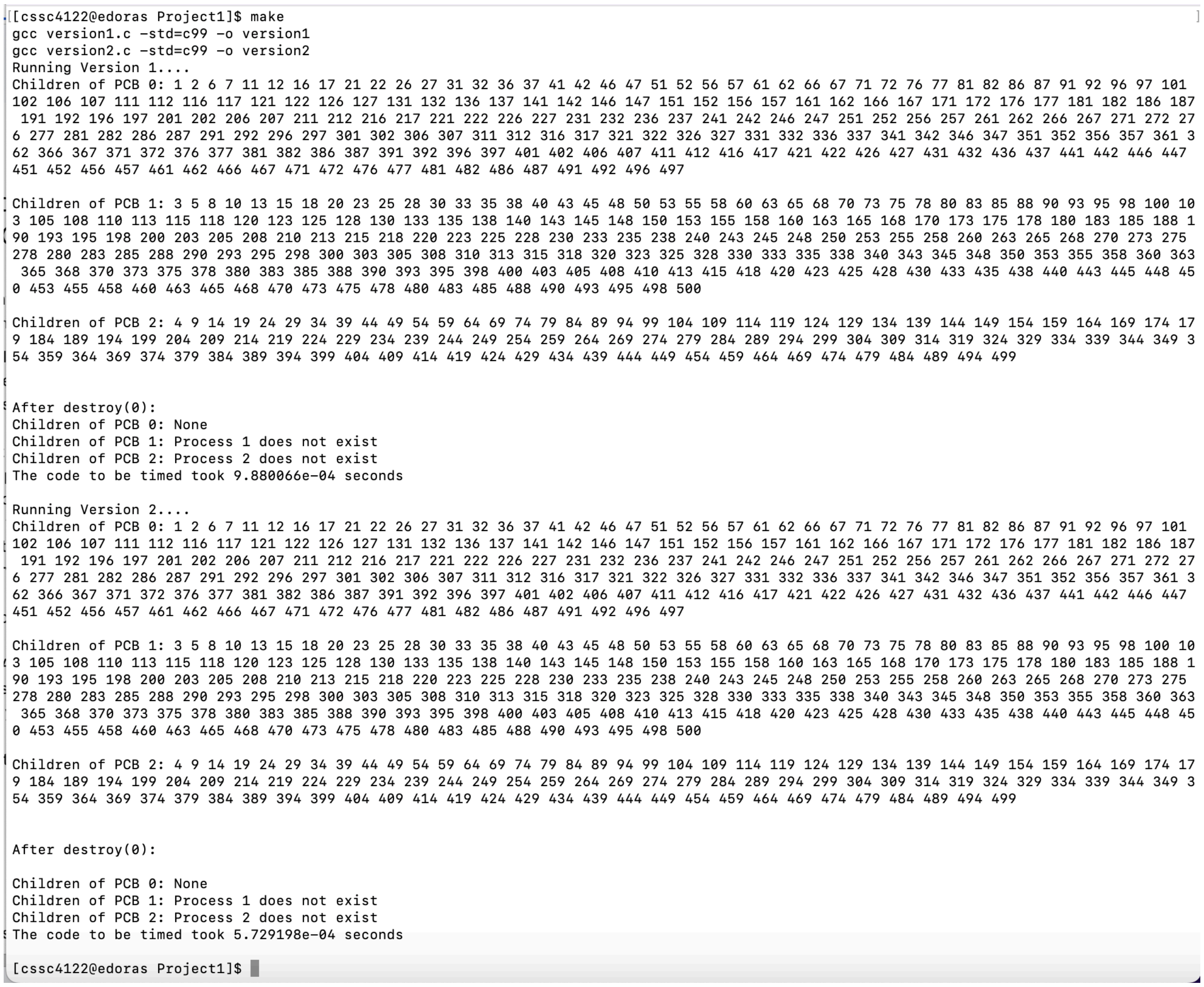
```
for (int i = 1; i <= 100; i++){
  create(0);
  create(0);
  create(1);
  create(2);
  create(1);
}

print(0);
print(1);
print(2);

process_being_destroyed = 0;
destroy(0);

printf("\nAfter destroy(0): \n");
print(0);
print(1);
print(2);
```

…and obtained the following output after the makefile execution:



```
[cssc4122@edoras Project1]$ make
gcc version1.c -std=c99 -o version1
gcc version2.c -std=c99 -o version2
Running Version 1....
Children of PCB 0: 1 2 6 7 11 12 16 17 21 22 26 27 31 32 36 37 41 42 46 47 51 52 56 57 61 62 66 67 71 72 76 77 81 82 86 87 91 92 96 97 101
102 106 107 111 112 116 117 121 122 126 127 131 132 136 137 141 142 146 147 151 152 156 157 161 162 166 167 171 172 176 177 181 182 186 187
 191 192 196 197 201 202 206 207 211 212 216 217 221 222 226 227 231 232 236 237 241 242 246 247 251 252 256 257 261 262 266 267 271 272 27
6 277 281 282 286 287 291 292 296 297 301 302 306 307 311 312 316 317 321 322 326 327 331 332 336 337 341 342 346 347 351 352 356 357 361 3
62 366 367 371 372 376 377 381 382 386 387 391 392 396 397 401 402 406 407 411 412 416 417 421 422 426 427 431 432 436 437 441 442 446 447
451 452 456 457 461 462 466 467 471 472 476 477 481 482 486 487 491 492 496 497

Children of PCB 1: 3 5 8 10 13 15 18 20 23 25 28 30 33 35 38 40 43 45 48 50 53 55 58 60 63 65 68 70 73 75 78 80 83 85 88 90 93 95 98 100 10
3 105 108 110 113 115 118 120 123 125 128 130 133 135 138 140 143 145 148 150 153 155 158 160 163 165 168 170 173 175 178 180 183 185 188 1
90 193 195 198 200 203 205 208 210 213 215 218 220 223 225 228 230 233 235 238 240 243 245 248 250 253 255 258 260 263 265 268 270 273 275
278 280 283 285 288 290 293 295 298 300 303 305 308 310 313 315 318 320 323 325 328 330 333 335 338 340 343 345 348 350 353 355 358 360 363
 365 368 370 373 375 378 380 383 385 388 390 393 395 398 400 403 405 408 410 413 415 418 420 423 425 428 430 433 435 438 440 443 445 448 45
0 453 455 458 460 463 465 468 470 473 475 478 480 483 485 488 490 493 495 498 500

Children of PCB 2: 4 9 14 19 24 29 34 39 44 49 54 59 64 69 74 79 84 89 94 99 104 109 114 119 124 129 134 139 144 149 154 159 164 169 174 17
9 184 189 194 199 204 209 214 219 224 229 234 239 244 249 254 259 264 269 274 279 284 289 294 299 304 309 314 319 324 329 334 339 344 349 3
54 359 364 369 374 379 384 389 394 399 404 409 414 419 424 429 434 439 444 449 454 459 464 469 474 479 484 489 494 499


After destroy(0):
Children of PCB 0: None
Children of PCB 1: Process 1 does not exist
Children of PCB 2: Process 2 does not exist
The code to be timed took 9.880066e-04 seconds

Running Version 2....
Children of PCB 0: 1 2 6 7 11 12 16 17 21 22 26 27 31 32 36 37 41 42 46 47 51 52 56 57 61 62 66 67 71 72 76 77 81 82 86 87 91 92 96 97 101
102 106 107 111 112 116 117 121 122 126 127 131 132 136 137 141 142 146 147 151 152 156 157 161 162 166 167 171 172 176 177 181 182 186 187
 191 192 196 197 201 202 206 207 211 212 216 217 221 222 226 227 231 232 236 237 241 242 246 247 251 252 256 257 261 262 266 267 271 272 27
6 277 281 282 286 287 291 292 296 297 301 302 306 307 311 312 316 317 321 322 326 327 331 332 336 337 341 342 346 347 351 352 356 357 361 3
62 366 367 371 372 376 377 381 382 386 387 391 392 396 397 401 402 406 407 411 412 416 417 421 422 426 427 431 432 436 437 441 442 446 447
451 452 456 457 461 462 466 467 471 472 476 477 481 482 486 487 491 492 496 497

Children of PCB 1: 3 5 8 10 13 15 18 20 23 25 28 30 33 35 38 40 43 45 48 50 53 55 58 60 63 65 68 70 73 75 78 80 83 85 88 90 93 95 98 100 10
3 105 108 110 113 115 118 120 123 125 128 130 133 135 138 140 143 145 148 150 153 155 158 160 163 165 168 170 173 175 178 180 183 185 188 1
90 193 195 198 200 203 205 208 210 213 215 218 220 223 225 228 230 233 235 238 240 243 245 248 250 253 255 258 260 263 265 268 270 273 275
278 280 283 285 288 290 293 295 298 300 303 305 308 310 313 315 318 320 323 325 328 330 333 335 338 340 343 345 348 350 353 355 358 360 363
 365 368 370 373 375 378 380 383 385 388 390 393 395 398 400 403 405 408 410 413 415 418 420 423 425 428 430 433 435 438 440 443 445 448 45
0 453 455 458 460 463 465 468 470 473 475 478 480 483 485 488 490 493 495 498 500

Children of PCB 2: 4 9 14 19 24 29 34 39 44 49 54 59 64 69 74 79 84 89 94 99 104 109 114 119 124 129 134 139 144 149 154 159 164 169 174 17
9 184 189 194 199 204 209 214 219 224 229 234 239 244 249 254 259 264 269 274 279 284 289 294 299 304 309 314 319 324 329 334 339 344 349 3
54 359 364 369 374 379 384 389 394 399 404 409 414 419 424 429 434 439 444 449 454 459 464 469 474 479 484 489 494 499


After destroy(0):

Children of PCB 0: None
Children of PCB 1: Process 1 does not exist
Children of PCB 2: Process 2 does not exist
The code to be timed took 5.729198e-04 seconds

[cssc4122@edoras Project1]$
```