

## Results:

### 1. Task1

```
hello! I'm from child and my process id is 2228.  
My parent process id is 2223  
hi! I'm from parent and my process id is 2223[1] + Done  
/tmp/Microsoft-MIEngine-In-3huwqqlz.log" 1>"/tmp/Microsoft-
```

#### Task 1 a

```
Parent id: 2942  
child id: 2950  
  
Process details:  
Parent id: 2942  
child id: 2952  
  
Process details:  
Parent id: 664  
child id: 2953  
  
Process details:  
Parent id: 664  
child id: 2955  
  
Process details:  
  
Parent id: 2934  
Process details:  
child id: 2942  
Parent id: 664  
child id: 2951  
  
Process details:  
Parent id: 664  
child id: 2954  
  
Process details:  
Parent id: 664  
child id: 2956  
[1] + Done  
"/usr/bin/gdb" --interpreter=mi  
tmp/Microsoft-MIEngine-Out-3g0omu20.juk"  
suyog@suyog-VirtualBox:~/Desktop/OS lab$
```

There are eight process in total out of which seven are child. Their process id along with their parent id is shown as above.

2.

```
0
1
2
3
4
5
6
7
8
9
10
a b c d e f g h i j k l m n o p q r
s t u v w x y z [1] + Done "/usr/bin/gdb" --interpreter=mi -
-tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-ssf0weon.diu" 1>"/tmp/Microsoft-MIEngine-Out-fgue024j.iyw"
suyog@suyog-VirtualBox:~/Desktop/OS lab$
```

In this program, the parent executes its program and the DEL2 holds the screen and again child executes and holds screen from DEL1.

When we increase the delays DEL1 and DEL2 then we saw that the output of two process overlapped. This is due to non-uniform delays and both process are running at the same time.

3.

```
i'm the child, my process ID is 3601
I'm the child and my parent's ID is 3598
(after sleep)i'm the child, my process ID is 3601
(after sleep)Im the child and my parent's ID is 3598
[1] + Done "/usr/bin/gdb" --interpreter=mi -
tmp/Microsoft-MIEngine-Out-5dqe5cbg.rxx"
suyog@suyog-VirtualBox:~/Desktop/OS lab$
```

Here, the execution of fork() was successful so the value of pid will be 0 and hence only the statements under if with condition pid=0 is executed.

Adding the sleep statement delayed the output by some time . The child process is an orphan process and a new parent is given to it by the system.

4.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 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
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 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
```

```

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 [1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${Dbg
m) 0<" /tmp/Microsoft-MIEngine-In-f433psa5_ybe" 1>" /tmp/Microsoft-MIEngine-Out-xaomumgw.v5k"
suyog@suyog-VirtualBox:~/Desktop/OS Lab$ []

```

In this program, first the parent executed and then the execution of the child took place.

After edit

```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 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 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 2
9 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 7
4 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 11
4 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 1
48 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181
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 24
9 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 2
83 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316
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 38
4 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 4
18 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451
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[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTer
m) 0<" /tmp/Microsoft-MIEngine-In-5hf3h0eb_ou5" 1>" /tmp/Microsoft-MIEngine-Out-3dai03zd.2rb"
suyog@suyog-VirtualBox:~/Desktop/OS Lab$

```

When the wait(0) statement is added, then the parent waits for its child to complete its execution and the executes itself. Only the parent can wait for the child and not the other way around.

**Discussion:**

In this lab session, we developed the concept of parent and child process. We used the fork statement to observe the behavior of parent and child process under different circumstances. We also observed the process id's using getpid() and getppid() statement. Also, the behavior of the processes under various delay conditions were also observed. Along with these, we also used the wait() statement to make the parent process wait till the completion of child process.

**Conclusion:**

In concluding, we became familiar with various process concepts and manipulation metrics for different processes. After this lab session we are able to understand the concept of parent and child process as well as the importance of proper delays for these processes.