OSL – WEEK 3

Q1)

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() { printf("[\*] Starting parent process\n");

int r\_val = fork(); switch(r\_val) { case -1: { perror("[\*] fork\n"); exit(EXIT\_FAILURE);

} break; case 0: { printf("[\*\*] Inside child process\n"); for(int i = 0; i < 10; i++) { sleep(1);

printf("[\*\*] Sleeping child for %d seconds...\

n", i + 1);

} exit(0);

} break; default: { printf("[\*] Waiting for child\n"); wait(NULL);

printf("[\*] Child process finished execution\n"); exit(0);

}

}

}



Q2)

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() { printf("[\*] Starting parent process\n"); int r\_val = fork(); switch(r\_val) { case -1: { perror("[\*] fork\n"); exit(EXIT\_FAILURE);

} break; case 0: { printf("[\*\*] Inside child process\n"); printf("[\*\*] Executing ./wait inside child\n"); execl("./wait", "./wait" , NULL);

exit(0);

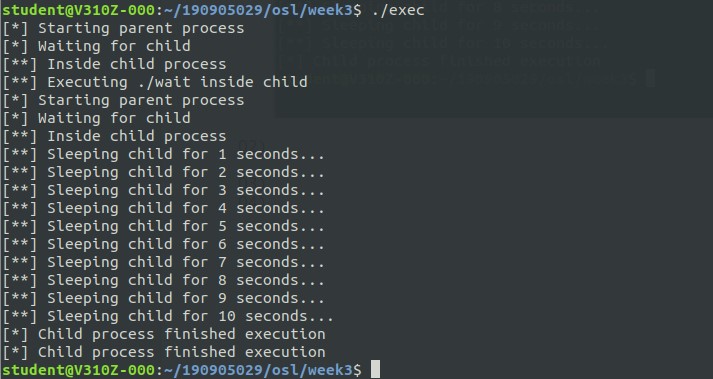
} break; default: { printf("[\*] Waiting for child\n"); wait(NULL);

printf("[\*] Child process finished execution\n"); exit(0);

}

}

}



Q3)

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() { printf("[\*] Starting parent process\n"); int r\_val = fork(); switch(r\_val) { case -1: { perror("[\*] fork\n"); exit(EXIT\_FAILURE);

} break; case 0: { printf("[\*\*] Inside child process\n");

pid\_t curr\_pid = getpid();

printf("[\*\*] PID of child process: %d\n", curr\_pid);

exit(0);

} break; default: { pid\_t curr\_pid = getpid();

printf("[\*] PID of parent process: %d\n", curr\_pid);

printf("[\*] Waiting for child\n"); wait(NULL);

printf("[\*] Child process finished execution\n"); exit(0);

}

}

}



Q4)

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() { printf("[\*] Starting parent process\n"); int r\_val = fork(); switch(r\_val) { case -1: { perror("[\*] fork\n"); exit(EXIT\_FAILURE);

} break; case 0: { printf("[\*\*] Inside child process\n");

pid\_t curr\_ppid = getppid();

printf("[\*\*] Current parent: %d\n", curr\_ppid);

for(int i = 0; i < 10; i++) { sleep(1);

printf("[\*\*] Sleeping child for %d seconds...\

n", i + 1);

}

curr\_ppid = getppid();

printf("[\*\*] Current parent: %d\n", curr\_ppid); exit(0);

} break; default: { printf("[\*\*] Executing ps\n"); execl("/bin/ps", "ps", NULL); exit(0);

}

}

}

