Στην εργασία 2 έχουν υλοποιηθεί τα 2 syscalls: int setpriority(int), int getpinfo(struct pstat*) η user level εντολή ps, καθώς και το priority scheduler,συνεπώς η εργασία
έχει ολοκληρωθεί

Δομή repository:
Η δομή του repository έχει μείνει ίδια με κάποιες μικρές προσθήκες στα απαραίτητα αρχεία για να αναγνωρίζονται τα syscalls από το user-level. Αυτές οι προσθήκες είναι:
.1 defs.h (πρόσθεσα τα syscalls στη περιοχή του proc.c)
.2 user.h(προσθήκη των syscalls στη περιοχή που τους αναλογεί)
.3 sysproc.c (έχουν προστεθεί οι υλοποιήσεις sys_setpriority(int) και sys_getpinfo()
.4 syscall.h ('Εγιναν define τα 2 syscalls)
.5 δηλώθηκαν οι συναρτήσεις με το κατάλληλο format.
.6 'Εγινε αλλαγή στο procinit() για να διασφαλιθεί το default priority κάθε μεταβλητής να ισούται με 10.
.7 'Εγινε αλλαγή στο usys.pl

Πέρα από αυτές τις αλλαγές δημιουργήθηκε το user-level αρχείο ps.c το οποίο θα αναλυθεί παραπάνω αργότερα.

int setpriority(int x):
Το syscall υλοποιείται στο proc.c. Δέχεται τον ακέραιο χ (έχουμε φροντίσει με τη συνάρτηση argaddr στο sys_setpriority) και ελέγχει για το αν βρίσκεται στο εύρος του 1-20. Αν ναι δηλώνει το priority της τώρινής διεργασίες ίσο με το x και επιστρέφει 0, αλλιώς δεν γίνεται κάποια δήλωση και επιστρέφει -1.

int getpinfo(struct pstat* statuses):
Αρχικά ορίστηκε το struct statuses στο proc.h με τα βασικά.7 χαρακτηριστικά που ορίζουν τη κάθε διεργασία (και υπήρχαν ήδη στο struct proc). Στη συνάρτηση getpinfo ελέγχουμε για κάθε διεργασία το

state της και αν θεωρηθεί ενεργή παιρνάμε τις μεταβλητές που υπάρχουν στο struct της στις ανάλογες μεταβλητές του τοπικού struct που έχουμε δημιουργήσει. Έπειτα χρησιμοποιύμε την συνάρτηση copyout() για να αντιγράψουμε όλο το pstat struct από το τοπικό kernel level struct στη μεταβλητή statuses που δόθηκε από το user-level.

ps.c:
Στο ps.c γίνεται επίδειξη της ορθής λειτουργίας των παραπάνω syscalls.Καλέιται ηsetpriority(14) καθιστώντας το priority του ps ίσο με 14, καθώς επίσης καλείται και η getpinfo παιρνώντας τα απαραίτητα χαρακτηριστκά κάθε ενεργής διεργασίας στο struct pstat* statuses. Τέλος για κάθε ενεργή διεργασία εκτυπώνει τα χαρακτηριστικά της μέσω της μεταβλητής pstat* statuses που δηλώθηκε στο user-level αρχείο.

Priority scheduler: Αρχικά ελέγχουμε ποια από όλες τις διεργασίες έχει το μεγαλύτερο priority. Έπειτα, μόλις τη βρούμε την αποθηκεύουμε στο τοπική μεταβλητή struct proc* που έχουμε δημιουργήσει. Έπειτα, ενώ έχουμε πάρει τα κατάλληλα μέτρα για αποφυγή race conditions (&high_priority_process->lock, καθώς και τη συνθήκη που ελέγχει αν το state παρέμεινε runnable) εκτελούμε τη διεργασία
με τον ίδιο ακριβώς τρόπο που συνέβαινε και στον default scheduler.